Chapter 20
Computationally Intensive Techniques

It is generally accepted that training in statistics must include some exposure to the mechanics of computational statistics. This exposure to computational methods is of an essential nature when we consider extremely high-dimensional data. Computer-aided techniques can help us to discover dependencies in high dimensions without complicated mathematical tools. A draftman's plot (i.e. a matrix of pairwise scatterplots like in Fig. 1.14) may lead us immediately to a theoretical hypothesis (on a lower dimensional space) on the relationship of the variables. Computer-aided techniques are therefore at the heart of multivariate statistical analysis.

With the rapidly increasing amount of data statistics faces a new challenge. While in the twentieth century the focus was on the mathematical precision of statistical modelling, the twenty-first century relies more and more on data analytic procedures that provide information (even for extremely large data bases) on the fingertip. This demand on fast availability of condensed statistical information has changed the statistical paradigm and has shifted energy from mathematical analysis to computational analysis of course without loosing sight of the statistical core questions.

In this chapter we first present the concept of Simplicial Depth—a multivariate extension of the data depth concept of Sect. 1.1. We then present Projection Pursuit—a semiparametric technique which is based on a one-dimensional, flexible regression or on the idea of density smoothing applied to principal component analysis (PCA) type projections. A similar model is underlying the Sliced Inverse Regression (SIR) technique which we discuss in Sect. 20.3.

The next technique is called support vector machines (SVMs) and is motivated by non-linear classification (discrimination) problems. SVMs are classification methods based on statistical learning theory. A quadratic optimisation problem determines so-called support vectors with high margin that guarantee maximal separability. Non-linear classification is achieved by mapping the data into a feature space and finding a linear separating hyperplane in this feature space. Another

W.K. Härdle, L. Simar, *Applied Multivariate Statistical Analysis*,
DOI 10.1007/978-3-662-45171-7_20

advanced technique is CART—Classification and Regression Trees, a decision tree procedure developed by Breiman, Friedman, Olshen, and Stone (1984).

20.1 Simplicial Depth

Simplicial depth generalises the notion of data depth as introduced in Sect. 1.1. This general definition allows us to define a multivariate median and to visually present high-dimensional data in low dimension. For univariate data we have well known parameters of location which describe the centre of a distribution of a random variable X. These parameters are for example the *mean*

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{20.1}$$

or the *mode*

$$x_{\text{mod}} = \arg\max_x \hat{f}(x),$$

where $\hat{f}$ is the estimated density function of X (see Sect. 1.3). The *median*

$$x_{\text{med}} = \begin{cases} x_{\{(n+1)/2\}} & \text{if } n \text{ odd} \\ \frac{x_{(n/2)} + x_{(n/2+1)}}{2} & \text{otherwise,} \end{cases}$$

where $x_{(i)}$ is the order statistics of the n observations x_i, is yet another measure of location.

The first two parameters can be easily extended to multivariate random variables. The mean in higher dimensions is defined as in (20.1) and the mode accordingly,

$$x_{\text{mod}} = \arg\max_x \hat{f}(x)$$

with $\hat{f}$ the estimated multidimensional density function of X (see Sect. 1.3). The median poses a problem though since in a multivariate sense we cannot interpret the element-wise median

$$x_{\text{med},j} = \begin{cases} x_{\{(n+1)/2\},j} & \text{if } n \text{ odd} \\ \frac{x_{(n/2),j} + x_{(n/2+1),j}}{2} & \text{otherwise} \end{cases} \tag{20.2}$$

as a point that is "most central". The same argument applies to other observations of a sample that have a certain "depth" as defined in Sect. 1.1. The "fourths" or the

"extremes" are not defined in a straightforward way in higher (not even for two) dimensions.

An equivalent definition of the median in one dimension is given by the *simplicial depth*. It is defined as follows: For each pair of datapoints x_i and x_j we generate a closed interval, a one-dimensional simplex, which contains x_i and x_j as border points. Redefine the median as the datapoint x_{med}, which is enclosed in the maximum number of intervals:

$$x_{\text{med}} = \arg\max_i \#\{k, l; x_i \in [x_k, x_l]\}. \tag{20.3}$$

With this definition of the median, the median is the "deepest" and "most central" point in a data set as discussed in Sect. 1.1. This definition involves a computationally intensive operation since we generate $n(n-1)/2$ intervals for n observations.

In two dimensions, the computation is even more intensive since the interval $[x_k, x_l]$ is replaced by a triangle constructed from three different datapoints. The median as the deepest point is then defined by that datapoint that is covered by the maximum number of triangles. In three dimensions triangles become pyramids formed from 4 points and the median is that datapoint that lies in the maximum number of pyramids.

An example for the depth in two dimensions is given by the constellation of points given in Fig. 20.1. If we build for example the triangle of the points 1, 3, 5 (denoted as $\triangle$ 135 in Table 20.1), it contains the point 4. From Table 20.1 we count the number of coverages to obtain the simplicial depth values of Table 20.2.

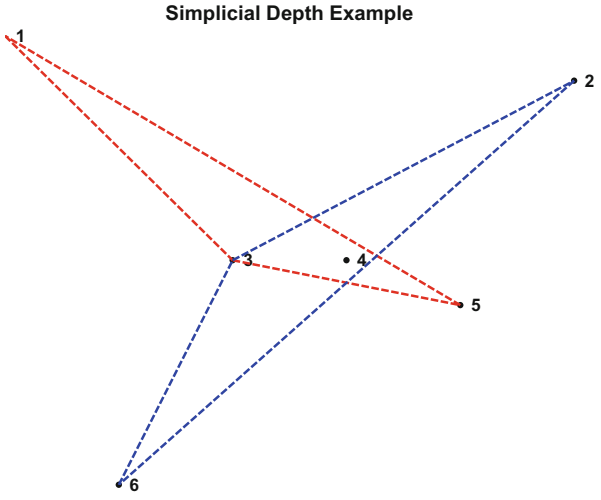

Fig. 20.1 Construction of simplicial depth `MVAsimdep1`

Table 20.1 Coverages for artificial configuration of points

	Triangle	Coverages					
1	△ 123	1	2	3			
2	△ 124	1	2		4		
3	△ 125	1	2			5	
4	△ 126	1	2	3	4		6
5	△ 134	1		3	4		
6	△ 135	1		3	4	5	
7	△ 136	1		3			6
8	△ 145	1			4	5	
9	△ 146	1		3	4		6
10	△ 156	1		3	4	5	6
11	△ 234		2	3	4		
12	△ 235		2	3	4	5	
13	△ 236		2	3	4		6
14	△ 245		2		4	5	
15	△ 246		2		4		6
16	△ 256		2			5	6
17	△ 345			3	4	5	
18	△ 346			3	4		6
19	△ 356			3		5	6
20	△ 456				4	5	6

Table 20.2 Simplicial depths for artificial configuration of points

Point	1	2	3	4	5	6
Depth	10	10	12	14	8	8

In arbitrary dimension p, we look for datapoints that lie inside a simplex (or convex *hull*) formed from $p + 1$ points. We therefore extend the definition of the median to the multivariate case as follows

$$x_{\text{med}} = \arg\max_i \#\{k_0, \ldots, k_p; x_i \in hull(x_{k_0}, \ldots, x_{k_p})\}. \tag{20.4}$$

Here $k_0, \ldots, k_p$ denote the indices of $p + 1$ datapoints. Thus for each datapoint we have a multivariate data depth. If we compute all the necessary simplices $hull(x_{k_0}, \ldots, x_{k_p})$, the computing time will unfortunately be exponential as the dimension increases.

In Fig. 20.2 we calculate the simplicial depth for a two-dimensional, 10 point distribution according to depth. It contains 100 data points with corresponding parameters controlling its spread. The deepest point, the two-dimensional median, is indicated as a big star in the centre. The points with less depth are indicated via grey shades.

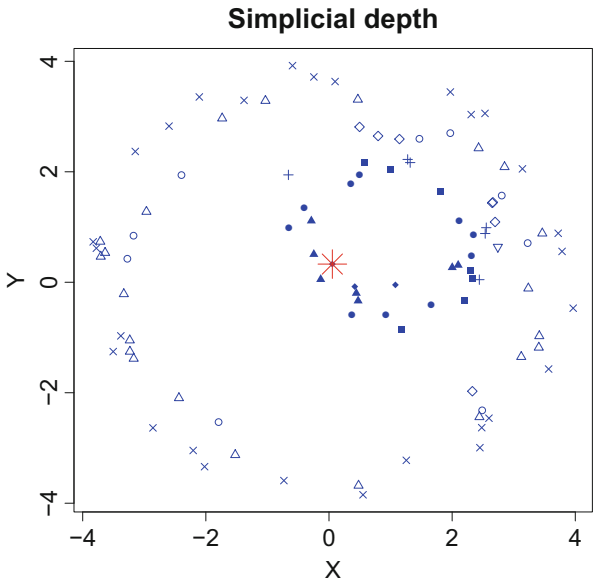

Fig. 20.2 10 point distribution according to depth with the median shown as a *big star* in the centre `MVAsimdepex`

Summary

- ↪ The "depth" of a datapoint in one dimension can be computed by counting all (closed) intervals of two datapoints which contain the datapoint
- ↪ The "deepest" datapoint is the central point of the distribution, the median
- ↪ The "depth" of a datapoint in arbitrary dimension p is defined as the number of simplices (constructed from $p + 1$ points) covering this point. It is called simplicial depth
- ↪ A multivariate extension of the median is to take the "deepest" datapoint of the distribution
- ↪ In the bivariate case we count all triangles of datapoints which contain the datapoint to compute its depth

20.2 Projection Pursuit

"Projection Pursuit" stands for a class of exploratory projection techniques. This class contains statistical methods designed for analysing high-dimensional data using low-dimensional projections. The aim of projection pursuit is to

reveal possible non-linear and therefore interesting structures hidden in the high-dimensional data. To what extent these structures are "interesting" is measured by an index. Exploratory Projection Pursuit (EPP) goes back to Kruskal (1969, 1972). The approach was successfully implemented for exploratory purposes by various other authors. The idea has been applied to regression analysis, density estimation, classification and discriminant analysis.

Exploratory Projection Pursuit

In EPP, we try to find "interesting" low-dimensional projections of the data. For this purpose, a suitable index function $I(\alpha)$, depending on a normalised projection vector α, is used. This function will be defined such that "interesting" views correspond to local and global maxima of the function. This approach naturally accompanies the technique of PCA of the covariance structure of a random vector X. In PCA we are interested in finding the axes of the covariance ellipsoid. The index function $I(\alpha)$ is in this case the variance of a linear combination $\alpha^\top X$ subject to the normalising constraint $\alpha^\top \alpha = 1$ (see Theorem 11.2). If we analyse a sample with a p-dimensional normal distribution, the "interesting" high-dimensional structure we find by maximising this index is of course linear.

There are many possible projection indices, for simplicity the kernel based and polynomial based indices are reported. Assume that the p-dimensional random variable X is sphered and centred, that is, $\mathsf{E}(X) = 0$ and $\mathsf{Var}(X) = \mathcal{I}_p$. This will remove the effect of location, scale, and correlation structure. This covariance structure can be achieved easily by the Mahalanobis transformation (3.26).

Friedman and Tukey (1974) proposed to investigate the high-dimensional distribution of X by considering the index

$$I_{\mathrm{FT},h}(\alpha) = n^{-1} \sum_{i=1}^{n} \hat{f}_{h,\alpha}(\alpha^\top X_i) \tag{20.5}$$

where $\hat{f}_{h,\alpha}$ denotes the kernel estimator (see Sect. 1.3)

$$\hat{f}_{h,\alpha}(z) = n^{-1} \sum_{j=1}^{n} K_h(z - \alpha^\top X_j) \tag{20.6}$$

of the projected data. Note that (20.5) is an estimate of $\int f^2(z)dz$ where $z = \alpha^\top X$ is a one-dimensional random variable with mean zero and unit variance. If the high-dimensional distribution of X is normal, then each projection $z = \alpha^\top X$ is standard normal since $||\alpha|| = 1$ and since X has been centred and sphered by, e.g. the Mahalanobis transformation.

The index should therefore be stable as a function of α if the high-dimensional data is in fact normal. Changes in $I_{\mathrm{FT},h}(\alpha)$ with respect to α therefore indicate deviations from normality. Hodges and Lehman (1956) showed that, given a mean of zero and unit variance, the (compact support) density which minimises $\int f^2$ is uniquely given by

$$f(z) = \max\{0, c(b^2 - z^2)\},$$

where $c = 3/(20\sqrt{5})$ and $b = \sqrt{5}$. This is a parabolic density function, which is equal to zero outside the interval $(-\sqrt{5}, \sqrt{5})$. A high value of the Friedman–Tukey index indicates a larger departure from the parabolic form.

An alternative index is based on the negative of the entropy measure, i.e. $\int -f \log f$. The density for zero mean and unit variance which minimises the index

$$\int f \log f$$

is the standard normal density, a far more plausible candidate than the parabolic density as a norm from which departure is to be regarded as "interesting". Thus in using $\int f \log f$ as a projection index we are really implementing the viewpoint of seeing "interesting" projections as departures from normality. Yet another index could be based on the Fisher information (see Sect. 6.2)

$$\int (f')^2/f.$$

To optimise the entropy index, it is necessary to recalculate it at each step of the numerical procedure. There is no method of obtaining the index via summary statistics of the multivariate data set, so the workload of the calculation at each iteration is determined by the number of observations. It is therefore interesting to look for approximations to the entropy index. Jones and Sibson (1987) suggested that deviations from the normal density should be considered as

$$f(x) = \varphi(x)\{1 + \varepsilon(x)\} \tag{20.7}$$

where the function ε satisfies

$$\int \varphi(u)\varepsilon(u)u^{-r}du = 0, \text{ for } r = 0, 1, 2. \tag{20.8}$$

In order to develop the Jones and Sibson (1987) index it is convenient to think in terms of cumulants $\kappa_3 = \mu_3 = \mathsf{E}(X^3)$, $\kappa_4 = \mu_4 = \mathsf{E}(X^4) - 3$ (see Sect. 1.3). The standard normal density satisfies $\kappa_3 = \kappa_4 = 0$, an index with any hope of tracking the entropy index must at least incorporate information up to the level of symmetric departures (κ_3 or κ_4 not zero) from normality. The simplest of such indices is a

positive definite quadratic form in κ_3 and κ_4. It must be invariant under sign-reversal of the data since both $\alpha^\top X$ and $-\alpha^\top X$ should show the same kind of departure from normality. Note that κ_3 is odd under sign-reversal, i.e. $\kappa_3(\alpha^\top X) = -\kappa_3(-\alpha^\top X)$. The cumulant κ_4 is even under sign-reversal, i.e. $\kappa_4(\alpha^\top X) = \kappa_4(-\alpha^\top X)$. The quadratic form in κ_3 and κ_4 measuring departure from normality cannot include a mixed $\kappa_3\kappa_4$ term.

For the density (20.7) one may conclude with (20.8) that

$$\int f(u)\log(u)du \approx \frac{1}{2}\int \varphi(u)\varepsilon(u)du.$$

Now if f is expressed as a Gram–Charliér expansion

$$f(x)\varphi(x) = \{1 + \kappa_3 H_3(x)/6 + \kappa_4 H_4(x)/24 + \cdots\} \tag{20.9}$$

(Kendall & Stuart, 1977, p. 169) where H_r is the r-th Hermite polynomial, then the truncation of (20.9) and use of orthogonality and normalisation properties of Hermite polynomials with respect to φ yields

$$\frac{1}{2}\int \varphi(x)\varepsilon^2(x)dx = \left(\kappa_3^2 + \kappa_4^2/4\right)/12.$$

The index proposed by Jones and Sibson (1987) is therefore

$$I_{\mathrm{JS}}(\alpha) = \{\kappa_3^2(\alpha^\top X) + \kappa_4^2(\alpha^\top X)/4\}/12.$$

This index measures in fact the negative entropy difference $\int f \log f - \int \varphi \log \varphi$.

Example 20.1 The EPP is used on the Swiss bank note data. For 50 randomly chosen one-dimensional projections of this six-dimensional dataset we calculate the Friedman–Tukey index to evaluate how "interesting" their structures are.

Figure 20.3 shows the density for the standard, normally distributed data (green) and the estimated densities for the best (red) and the worst (blue) projections found. A dotplot of the projections is also presented. In the lower part of the figure we see the estimated value of the Friedman–Tukey index for each computed projection. From this information we can judge the non normality of the bank note data set since there is a lot of variation across the 50 random projections.

Projection Pursuit Regression

The problem in projection pursuit regression is to estimate a response surface

$$f(x) = \mathsf{E}(Y \mid x)$$

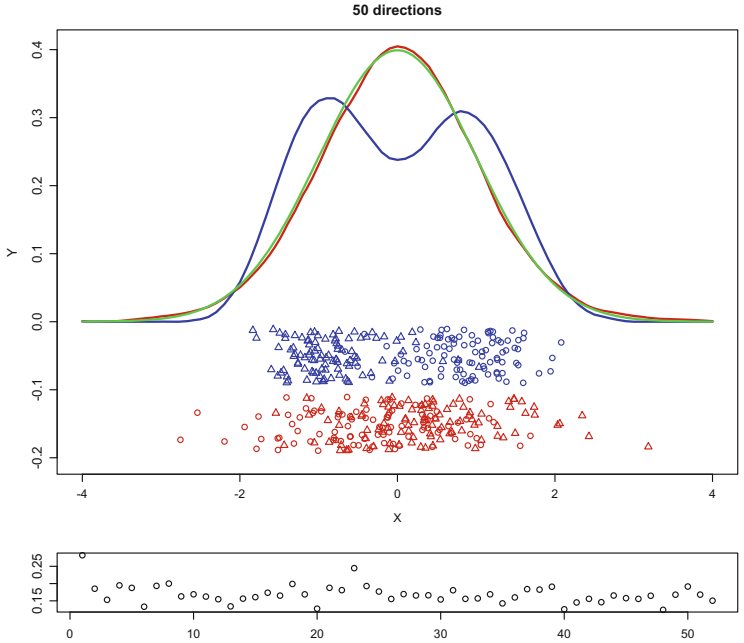

Fig. 20.3 Exploratory Projection Pursuit for the Swiss bank notes data (*green* = standard normal, *red* = best, *blue* = worst) `MVAppexample`

via approximating functions of the form:

$$\hat{f}(x) = \sum_{k=1}^{M} g_k(\Lambda_k^\top x)$$

with non-parametric regression functions g_k and projection indices Λ_k. Given observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ the basic algorithm works as follows.

1. Set $r_i^{(0)} = y_i$ and $k = 1$.
2. Minimise

$$E_k = \sum_{i=1}^{n} \left\{ r_i^{(k-1)} - g_k(\Lambda_k^\top x_i) \right\}^2$$

   where Λ_k is an orthogonal projection matrix and g_k is a non-parametric regression estimator.
3. Compute new residuals

$$r_i^{(k)} = r_i^{(k-1)} - g_k(\Lambda_k^\top x_i).$$

4. Increase k and repeat the last two steps until E_k becomes small.

Although this approach seems to be simple, we encounter some problems. One of the most serious is that the decomposition of a function into sums of functions of projections may not be unique. An example is

$$z_1 z_2 = \frac{1}{4ab}\{(az_1 + bz_2)^2 - (az_1 - bz_2)^2\}.$$

Numerical improvements of this algorithm were suggested by Friedman and Stuetzle (1981).

Summary

↪ Exploratory Projection Pursuit is a technique used to find interesting structures in high-dimensional data via low-dimensional projections. Since the Gaussian distribution represents a standard situation, we define the Gaussian distribution as the most uninteresting

↪ The search for interesting structures is done via a projection score like the Friedman–Tukey index $I_{\mathrm{FT}}(\alpha) = \int f^2$. The parabolic distribution has the minimal score. We maximise this score over all projections

↪ The Jones–Sibson index maximises

$$I_{\mathrm{JS}}(\alpha) = \{\kappa_3(\alpha^\top X) + \kappa_4^2(\alpha^\top X)/4\}/12$$

as a function of α

↪ The entropy index maximises

$$I_{\mathrm{E}}(\alpha) = \int f(\alpha^\top X) \log f(\alpha^\top X)$$

where f is the density of $\alpha^\top X$

↪ In Projection Pursuit Regression the idea is to represent the unknown function by a sum of non-parametric regression functions on projections. The key problem is in choosing the number of terms and often the interpretability

20.3 Sliced Inverse Regression

SIR is a dimension reduction method proposed by Duan and Li (1991). The idea is to find a smooth regression function that operates on a variable set of projections. Given a response variable Y and a (random) vector $X \in \mathbb{R}^p$ of explanatory variables, SIR is based on the model:

$$Y = m(\beta_1^\top X, \ldots, \beta_k^\top X, \varepsilon), \tag{20.10}$$

where $\beta_1, \ldots, \beta_k$ are unknown projection vectors, k is unknown and assumed to be less than p, $m : \mathbb{R}^{k+1} \to \mathbb{R}$ is an unknown function, and ε is the noise random variable with $\mathsf{E}(\varepsilon \,|\, X) = 0$.

Model (20.10) describes the situation where the response variable Y depends on the p-dimensional variable X only through a k-dimensional subspace. The unknown β_i's, which span this space, are called *effective dimension reduction directions* (EDR-directions). The span is denoted as *effective dimension reduction space* (EDR-space). The aim is to estimate the base vectors of this space, for which neither the length nor the direction can be identified. Only the space in which they lie is identifiable.

SIR tries to find this k-dimensional subspace of $\mathbb{R}^p$ which under the model (20.10) carries the essential information of the regression between X and Y. SIR also focuses on small k, so that nonparametric methods can be applied for the estimation of m. A direct application of nonparametric smoothing to X is for high dimension p generally not possible due to the sparseness of the observations. This fact is well known as the *curse of dimensionality*, see Huber (1985).

The name of SIR comes from computing the inverse regression (IR) curve. That means instead of looking for $\mathsf{E}(Y \,|\, X = x)$, we investigate $\mathsf{E}(X \,|\, Y = y)$, a curve in $\mathbb{R}^p$ consisting of p one-dimensional regressions. What is the connection between the IR and the SIR model (20.10)? The answer is given in the following theorem from Li (1991).

Theorem 20.1 *Given the model (20.10) and the assumption*

$$\forall b \in \mathbb{R}^p : \mathsf{E}\left(b^\top X \,|\, \beta_1^\top X = \beta_1^\top x, \ldots, \beta_k^\top X = \beta_k^\top x\right) = c_0 + \sum_{i=1}^{k} c_i \beta_i^\top x, \tag{20.11}$$

the centred IR curve $\mathsf{E}(X \,|\, Y = y) - \mathsf{E}(X)$ *lies in the linear subspace spanned by the vectors* $\Sigma\beta_i$, $i = 1, \ldots, k$, *where* $\Sigma = \mathsf{Cov}(X)$.

Assumption (20.11) is equivalent to the fact that X has an elliptically symmetric distribution, see Cook and Weisberg (1991). Hall and Li (1993) have shown that assumption (20.11) only needs to hold for the EDR-directions.

It is easy to see that for the standardised variable $Z = \Sigma^{-1/2}\{X - \mathsf{E}(X)\}$ the IR curve $m_1(y) = \mathsf{E}(Z \mid Y = y)$ lies in $\text{span}(\eta_1, \ldots, \eta_k)$, where $\eta_i = \Sigma^{1/2}\beta_i$. This means that the conditional expectation $m_1(y)$ is moving in $\text{span}(\eta_1, \ldots, \eta_k)$ depending on y. With b orthogonal to $\text{span}(\eta_1, \ldots, \eta_k)$, it follows that

$$b^\top m_1(y) = 0,$$

and further that

$$m_1(y)m_1(y)^\top b = \mathsf{Cov}\{m_1(y)\}b = 0.$$

As a consequence $\mathsf{Cov}\{\mathsf{E}(Z \mid y)\}$ is degenerated in each direction orthogonal to all EDR-directions η_i of Z. This suggests the following algorithm.

First, estimate $\mathsf{Cov}\{m_1(y)\}$ and then calculate the orthogonal directions of this matrix (for example, with eigenvalue/eigenvector decomposition). In general, the estimated covariance matrix will have full rank because of random variability, estimation errors and numerical imprecision. Therefore, we investigate the eigenvalues of the estimate and ignore eigenvectors having small eigenvalues. These eigenvectors $\hat{\eta}_i$ are estimates for the EDR-direction η_i of Z. We can easily rescale them to estimates $\hat{\beta}_i$ for the EDR-directions of X by multiplying by $\hat{\Sigma}^{-1/2}$, but then they are not necessarily orthogonal. SIR is strongly related to PCA. If all of the data falls into a single interval, which means that $\widehat{\mathsf{Cov}}\{m_1(y)\}$ is equal to $\widehat{\mathsf{Cov}}(Z)$, SIR coincides with PCA. Obviously, in this case any information about y is ignored.

The SIR Algorithm

The algorithm to estimate the EDR-directions via SIR is as follows:

1. Standardise x:

$$z_i = \hat{\Sigma}^{-1/2}(x_i - \bar{x}).$$

2. Divide the range of y_i into S nonoverlapping intervals *(slices)* $H_s, s = 1, \ldots, S$. n_s denotes the number of observations within slice H_s, and $\boldsymbol{I}_{H_s}$ the indicator function for this slice:

$$n_s = \sum_{i=1}^{n} \boldsymbol{I}_{H_s}(y_i).$$

3. Compute the mean of z_i over all slices. This is a crude estimate $\hat{m}_1$ for the *inverse regression curve* m_1:

$$\bar{z}_s = {n_s}^{-1} \sum_{i=1}^{n} z_i \ \boldsymbol{I}_{H_s}(y_i).$$

4. Calculate the estimate for $\mathsf{Cov}\{m_1(y)\}$:

$$\hat{V} = n^{-1} \sum_{s=1}^{S} n_s \bar{z}_s \bar{z}_s^\top .$$

5. Identify the eigenvalues $\hat{\lambda}_i$ and eigenvectors $\hat{\eta}_i$ of $\hat{V}$.
6. Transform the standardised EDR-directions $\hat{\eta}_i$ back to the original scale. Now the estimates for the EDR-directions are given by

$$\hat{\beta}_i = \hat{\Sigma}^{-1/2} \hat{\eta}_i .$$

Remark 20.1 The number of different eigenvalues unequal to zero depends on the number of slices. The rank of $\hat{V}$ cannot be greater than the number of slices -1 (the z_i sum up to zero). This is a problem for categorical response variables, especially for a binary response—where only one direction can be found.

SIR II

In the previous section we learned that it is interesting to consider the IR curve, that is, $\mathsf{E}(X \mid y)$. In some situations however SIR does not find the EDR-direction. We overcome this difficulty by considering the conditional covariance $\mathsf{Cov}(X \mid y)$ instead of the IR curve. An example where the EDR directions are not found via the SIR curve is given below.

Example 20.2 Suppose that $(X_1, X_2)^\top \sim N(0, \mathcal{I}_2)$ and $Y = X_1^2$. Then $\mathsf{E}(X_2 \mid y) = 0$ because of independence and $\mathsf{E}(X_1 \mid y) = 0$ because of symmetry. Hence, the EDR-direction $\beta = (1, 0)^\top$ is not found when the IR curve $\mathsf{E}(X \mid y) = 0$ is considered.

The conditional variance

$$\mathsf{Var}(X_1 \mid Y = y) = \mathsf{E}(X_1^2 \mid Y = y) = y,$$

offers an alternative way to find β. It is a function of y while $\mathsf{Var}(X_2 \mid y)$ is a constant.

The idea of SIR II is to consider the conditional covariances. The principle of SIR II is the same as before: investigation of the IR curve (here the conditional covariance instead of the conditional expectation). Unfortunately, the theory of SIR II is more complicated. The assumption of the elliptical symmetrical distribution of X has to be more restrictive, i.e. assuming the normality of X.

Given this assumption, one can show that the vectors with the largest distance to $\mathsf{Cov}(Z \mid Y = y) - \mathsf{E}\{\mathsf{Cov}(Z \mid Y = y)\}$ for all y are the most interesting for the

EDR-space. An appropriate measure for the overall mean distance is, according to Li (1992),

$$\begin{aligned} &\mathsf{E}\left(||\left[\mathsf{Cov}(Z \mid Y = y) - \mathsf{E}\{\mathsf{Cov}(Z \mid Y = y)\}\right] b||^2\right) \\ &\quad = b^\top \mathsf{E}\left(||\,\mathsf{Cov}(Z \mid y) - \mathsf{E}\{\mathsf{Cov}(Z \mid y)\}||^2\right) b. \end{aligned} \tag{20.12}$$

Equipped with this distance, we conduct again an eigensystem decomposition, this time for the above expectation $\mathsf{E}\left(||\,\mathsf{Cov}(Z \mid y) - \mathsf{E}\{\mathsf{Cov}(Z \mid y)\}||^2\right)$. Then we take the rescaled eigenvectors with the largest eigenvalues as estimates for the unknown EDR-directions.

The SIR II Algorithm

The algorithm of SIR II is very similar to the one for SIR, it differs in only two steps. Instead of merely computing the mean, the covariance of each slice has to be computed. The estimate for the above expectation (20.12) is calculated after computing all slice covariances. Finally, decomposition and rescaling are conducted, as before.

1. Do steps 1–3 of the SIR algorithm.
2. Compute the slice covariance matrix $\hat{V}_s$:

$$\hat{V}_s = (n_s - 1)^{-1} \sum_{i=1}^{n} I_{H_s}(y_i) z_i z_i^\top - n_s \bar{z}_s \bar{z}_s^\top.$$

3. Calculate the mean over all slice covariances:

$$\bar{V} = n^{-1} \sum_{s=1}^{S} n_s \hat{V}_s.$$

4. Compute an estimate for (20.12):

$$\hat{V} = n^{-1} \sum_{s=1}^{S} n_s \left(\hat{V}_s - \bar{V}\right)^2 = n^{-1} \sum_{s=1}^{S} n_s \hat{V}_s^2 - \bar{V}^2.$$

5. Identify the eigenvectors and eigenvalues of $\hat{V}$ and scale back the eigenvectors. This gives estimates for the SIR II EDR-directions:

$$\hat{\beta}_i = \hat{\Sigma}^{-1/2} \hat{\eta}_i.$$

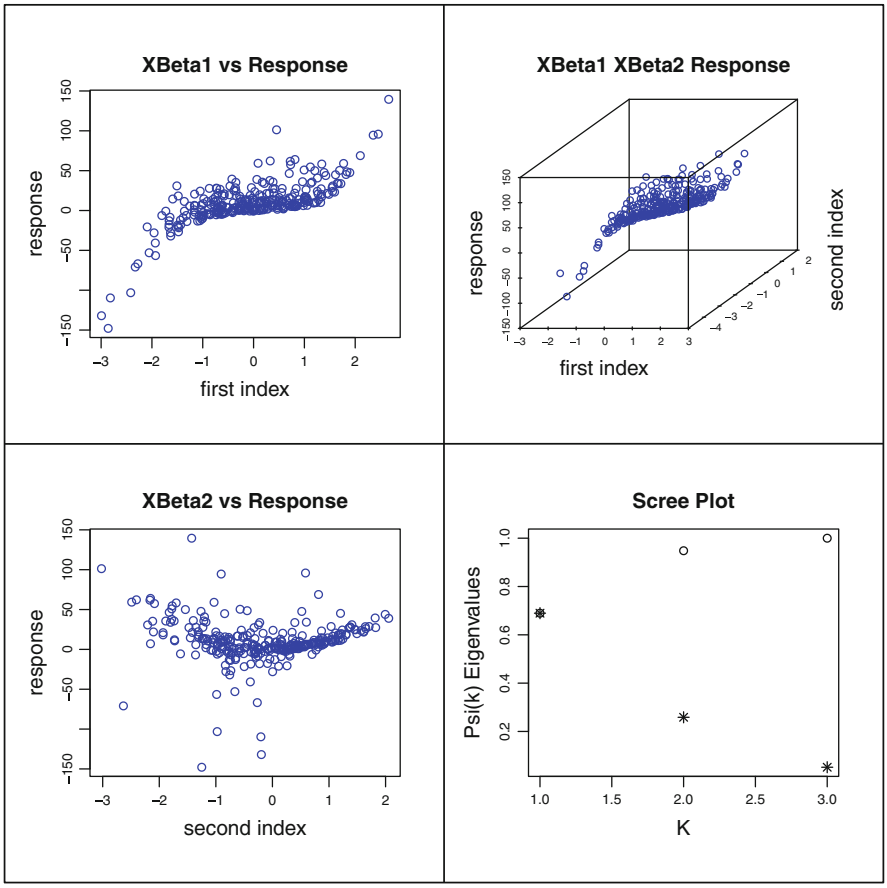

Fig. 20.4 SIR: The *left plots* show the response versus the estimated EDR-directions. The *upper right plot* is a three-dimensional plot of the first two directions and the response. The *lower right plot* shows the eigenvalues $\hat{\lambda}_i$ (*asterisk*) and the cumulative sum (*open circle*) MVAsirdata

Example 20.3 The result of SIR is visualised in four plots in Fig. 20.4: the left two show the response variable versus the first respectively second direction. The upper right plot consists of a three-dimensional plot of the first two directions and the response. The last picture shows $\hat{\Psi}_k$, the ratio of the sum of the first k eigenvalues and the sum of all eigenvalues, similar to PCA.

The data are generated according to the following model:

$$y_i = \beta_1^\top x_i + (\beta_1^\top x_i)^3 + 4\left(\beta_2^\top x_i\right)^2 + \varepsilon_i,$$

where the x_i's follow a three-dimensional normal distribution with zero mean, the covariance equal to the identity matrix, $\beta_2 = (1, -1, -1)^\top$, and $\beta_1 = (1, 1, 1)^\top$.

Fig. 20.5 Plot of the true response versus the true first index. The monotonic and the convex shapes can be clearly seen `MVAsirdata`

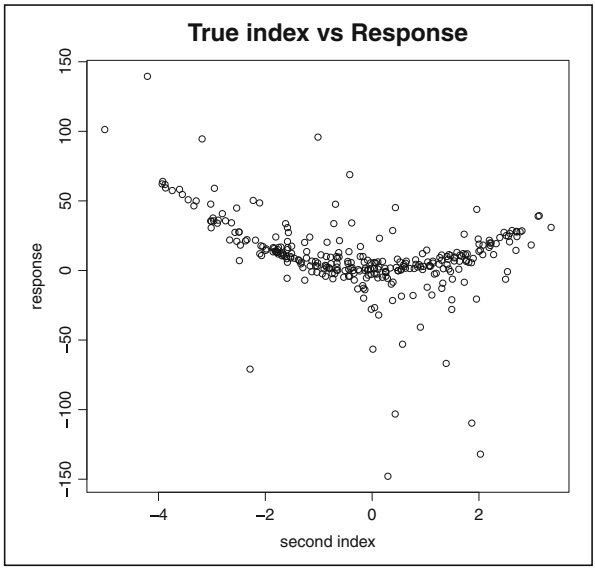

Fig. 20.6 Plot of the true response versus the true second index. The monotonic and the convex shapes can be clearly seen `MVAsirdata`

ε_i is standard, normally distributed and $n = 300$. Corresponding to model (20.10), $m(u, v, \varepsilon) = u + u^3 + v^2 + \varepsilon$. The situation is depicted in Figs. 20.5 and 20.6.

Both algorithms were conducted using the slicing method with 20 elements in each slice. The goal was to find β_1 and β_2 with SIR. The data are designed such that SIR can detect β_1 because of the monotonic shape of $\{\beta_1^\top x_i + (\beta_1^\top x_i)^3\}$, while SIR II will search for β_2, as in this direction the conditional variance on y is varying.

Table 20.3 SIR: EDR-directions for simulated data

$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$
0.452	0.881	0.040
0.571	−0.349	−0.787
0.684	−0.320	0.615

Table 20.4 SIR II: EDR-directions for simulated data

$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$
−0.272	0.964	−0.001
0.670	0.100	0.777
0.690	0.244	−0.630

If we normalise the eigenvalues for the EDR-directions in Table 20.3 such that they sum up to one, the resulting vector is $(0.852, 0.086, 0.062)$. As can be seen in the upper left plot of Fig. 20.4, there is a functional relationship found between the first index $\hat{\beta}_1^\top x$ and the response. Actually, β_1 and $\hat{\beta}_1$ are nearly parallel, that is, the normalised inner product $\hat{\beta}_1^\top \beta_1 / \{||\hat{\beta}_1|| ||\beta_1||\} = 0.9894$ is very close to one.

The second direction along β_2 is probably found due to the good approximation, but SIR does not provide it clearly, because it is "blind" with respect to the change of variance, as the second eigenvalue indicates.

For SIR II, the normalised eigenvalues are $(0.706, 0.185, 0.108)$, that is, about 69 % of the variance is explained by the first EDR-direction (Table 20.4). Here, the normalised inner product of β_2 and $\hat{\beta}_1$ is 0.9992. The estimator $\hat{\beta}_1$ estimates in fact β_2 of the simulated model. In this case, SIR II found the direction where the second moment varies with respect to $\beta_2^\top x$ (Fig. 20.7).

In summary, SIR has found the direction which shows a strong relation regarding the conditional expectation between $\beta_1^\top x$ and y, and SIR II has found the direction where the conditional variance is varying, namely, $\beta_2^\top x$.

The behaviour of the two SIR algorithms is as expected. In addition, we have seen that it is worthwhile to apply both versions of SIR. It is possible to combine SIR and SIR II (Cook & Weisberg, 1991; Li, 1991; Schott, 1994) directly, or to investigate higher conditional moments. For the latter it seems to be difficult to obtain theoretical results.

Summary

- ↪ SIR serves as a dimension reduction tool for regression problems
- ↪ Inverse regression avoids the *curse of dimensionality*
- ↪ The dimension reduction can be conducted without estimation of the regression function $y = m(x)$

Summary (continued)
↪ SIR searches for the effective dimension reduction (EDR) by computing the inverse regression IR
↪ SIR II uses the EDR on computing the inverse conditional variance
↪ SIR might miss EDR directions that are found by SIR II

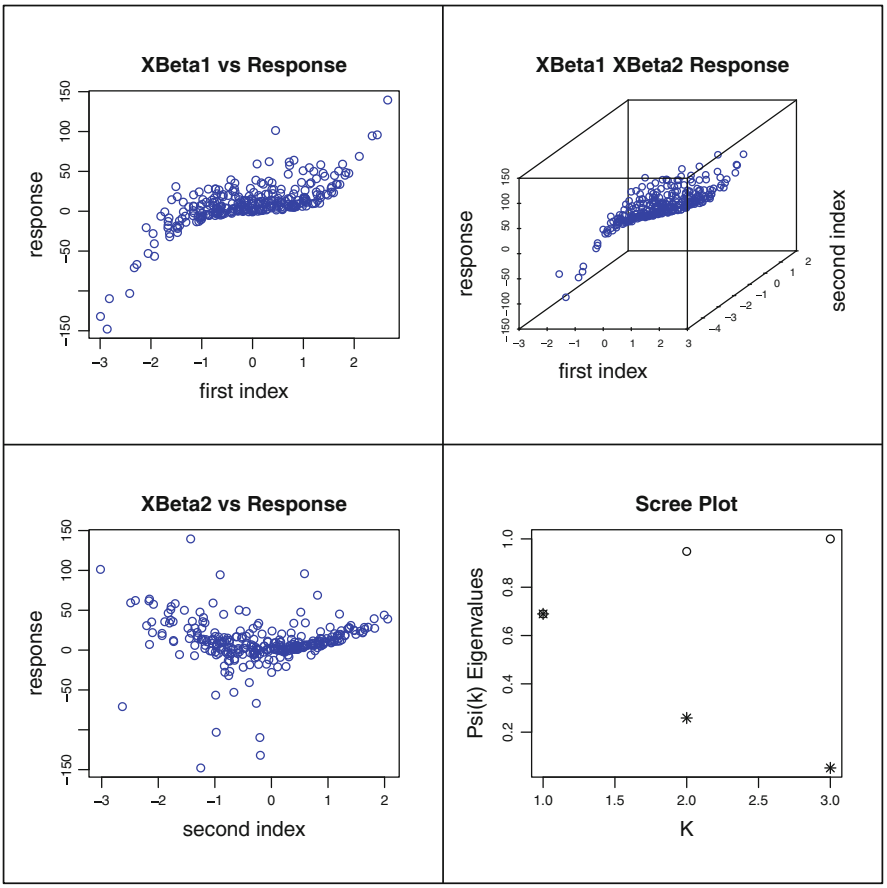

Fig. 20.7 SIR II mainly sees the direction β_2. The *left plots* show the response versus the estimated EDR-directions. The *upper right plot* is a three-dimensional plot of the first two directions and the response. The *lower right plot* shows the eigenvalues $\hat{\lambda}_i$ (*asterisk*) and the cumulative sum (*open circle*) `MVAsir2data`

20.4 Support Vector Machines

The purpose of this section is to introduce one of the most promising among recently developed multivariate non-linear statistical techniques: the SVM. The SVM is a classification method that is based on statistical learning theory. It has been successfully applied to optical character recognition, early medical diagnostics, and text classification. One application where SVMs outperformed other methods is electric load prediction (EUNITE, 2001), another one is optical character recognition (Vapnik, 1995). In a variety of applications SVMs produce better classification results than parametric methods (e.g. logit analysis) and are outperforming widely used nonparametric techniques, such as neural networks. Here we apply SVMs to corporate bankruptcy analysis.

Classification Methodology

In order to illustrate the classification methodology we focus for the moment on a company rating example that we will treat further in more detail. Investment risks are evaluated via the default probability (PD) for a company. Each company is described by a set of variables (predictors) x, such as financial ratios, and its class y that can be either $y = -1$ ("successful") or $y = 1$ ("bankrupt"). Financial ratios are constructed from the variables like net income, total assets, interest payments, etc. A training set represents a sample of data for companies which are known to have survived or gone bankrupt. From the training set one estimates a classifier function f that is then applied to computing PDs. These PDs can be uniquely translated into a company rating.

Classical discriminant analysis is based on the assumption that each group of observations is normally distributed with the same variance–covariance matrix but different means. Under such a formulation the discriminating function will be linear, see Theorem 14.2. Figure 20.8 displays this situation: if some linear combination of predictors (called Z-score in the context of bankruptcy analysis) is greater than a particular threshold value z_0 the observation under consideration is regarded as belonging to $y = 1$; if $Z < z_0$ the observation would belong to $y = -1$ (successful). One can change the labels "−1,+1" to the more standard notation "0,1". The current labeling is done only for mathematical convenience.

The Z-score is:

$$Z_i = a_1 x_{i1} + a_2 x_{i2} + \ldots + a_p x_{ip} = a^\top x_i,$$

where $x_i = (x_{i1}, \ldots, x_{ip})^\top \in \mathbb{R}^p$ are predictors for the i-th company. The classification based on the Z-score are necessarily linear and, therefore, may not handle more complex situations as in Fig. 20.9 when non-linear classifiers, such as those generated by SVMs, can produce better results.

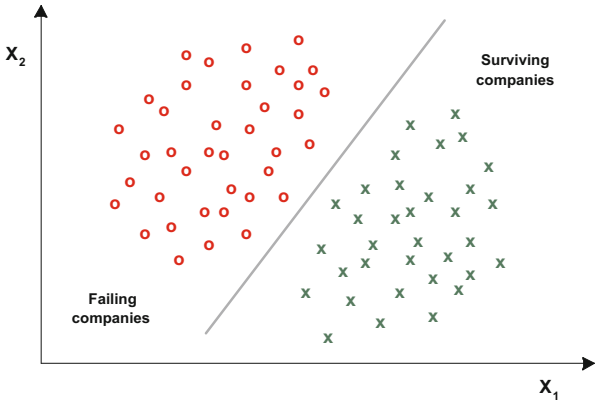

Fig. 20.8 A linear classification function in the case of linearly separable data

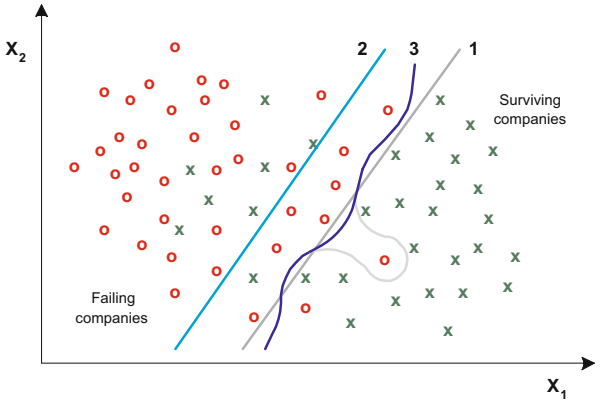

Fig. 20.9 Different linear classification functions (1) and (2) and a non-linear one (3) in the linearly non-separable case

Expected vs. Empirical Risk Minimisation

A non-linear classifier function f may be described by a function class $\mathcal{F}$. $\mathcal{F}$ is fixed a priori, e.g. it can be the class of linear classifiers (hyperplanes). A good classifier optimises some criterion that tells us how well f separates the classes. As in (14.4) one considers the minimisation of the expected risk:

$$R(f) = \int \frac{1}{2} |f(x) - y| \, dF(x, y). \tag{20.13}$$

The joint distribution $F(x, y)$, however, is never known in practical applications and must be estimated from the *training set* $\{x_i, y_i\}_{i=1}^n$. By replacing $F(x, y)$ with the empirical cdf $F_n(x, y)$ one obtains the empirical risk:

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} |f(x_i) - y_i| \, . \tag{20.14}$$

The empirical risk is an average value of loss over the training set, while the expected risk is the expected value of loss under the true probability measure. The loss is given by:

$$L(x, y) = \frac{1}{2}\,|f(x) - y| = \begin{cases} 0, & \text{if classification is correct,} \\ 1, & \text{if classification is wrong.} \end{cases}$$

One sees here that it is convenient to work with the labels "$-1, 1$" for y. The solutions to the problems of expected and empirical risk minimisation:

$$f_{\text{opt}} = \arg\min_{f \in \mathcal{F}} R\,(f)\,, \tag{20.15}$$

$$\hat{f}_n = \arg\min_{f \in \mathcal{F}} \hat{R}\,(f)\,, \tag{20.16}$$

generally do not coincide (Fig. 20.10), although converge as $n \to \infty$ if $\mathcal{F}$ is not too large. According to statistical learning theory (Vapnik, 1995), it is possible to get a uniform upper bound on the difference between $R\,(f)$ and $\hat{R}\,(f)$ via the Vapnik–Chervonenkis (VC) theory. The VC bound states that there is a function ϕ (monotone increasing in h) so that for all $f \in \mathcal{F}$ with a probability $1 - \eta$:

$$R\,(f) \leq \hat{R}\,(f) + \phi\left\{\frac{h}{n}, \frac{\log(\eta)}{n}\right\}. \tag{20.17}$$

Here h denotes the VC dimension, a measure of complexity of the involved function class $\mathcal{F}$. For a linear classification rule $g(x) = \text{sign}(x^\top w + b)$:

$$\phi\left\{\frac{h}{n}, \frac{\log(\eta)}{n}\right\} = \sqrt{\frac{h\left(\log\frac{2n}{h}\right) - \log\frac{\eta}{4}}{n}}, \tag{20.18}$$

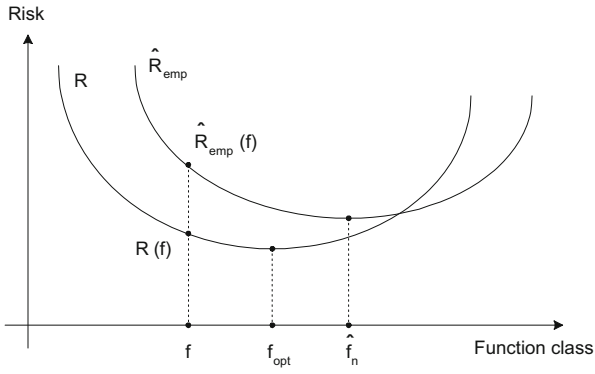

Fig. 20.10 The minima f_{opt} and $\hat{f}_n$ of the expected (R) and empirical ($\hat{R}$) risk functions generally do not coincide

where h is the VC dimension. By plotting the function $\phi(u, v) = \left\{ -u \cdot \log 2u + \log 4 - v \right\}^{-1/2}$ for small u one sees the monotonicity of $\phi(u, v)$. In fact one can show that

$$\frac{\partial \phi\left(\frac{h}{n}, \frac{\log(\eta)}{n}\right)}{\partial h} \geqslant 0$$

if and only if $2n \geqslant h$. For a linear classifier with $h = p + 1$ this is an easy condition to meet.

The VC dimension of a set $\mathcal{F}$ of functions in a d-dimensional space is h if some function $f \in \mathcal{F}$ can shatter h objects $\left\{x_i \in \mathbb{R}^d, i = 1, \ldots, h\right\}$, in all 2^h possible configurations and no set $\left\{x_j \in \mathbb{R}^d, j = 1, \ldots, q\right\}$ with $q > h$, exists that satisfies this property. For example, three points on a plane ($d = 2$) can be shattered by linear indicator functions in $2^h = 2^3 = 8$ ways, whereas 4 points can not be shattered in $2^q = 2^4 = 16$ ways. Thus, the VC dimension of the set of linear indicator functions in a two-dimensional space is $h = 3$, see Fig. 20.11. The expression for the VC bound (20.17) involves the VC dimension h, a parameter controlling complexity of $\mathcal{F}$. The term $\phi\left\{\frac{h}{n}, \frac{\log(\eta)}{n}\right\}$ introduces a penalty for excessive complexity of a classifier function. The higher is the complexity of $f \in \mathcal{F}$ the higher are h and therefore ϕ. There is a trade-off between the number of classification errors on the training set and the complexity of the classifier function. If the complexity were not controlled for, it would be possible to construct a classifier function with no classification errors on the training set notwithstanding how low its generalisation ability would be.

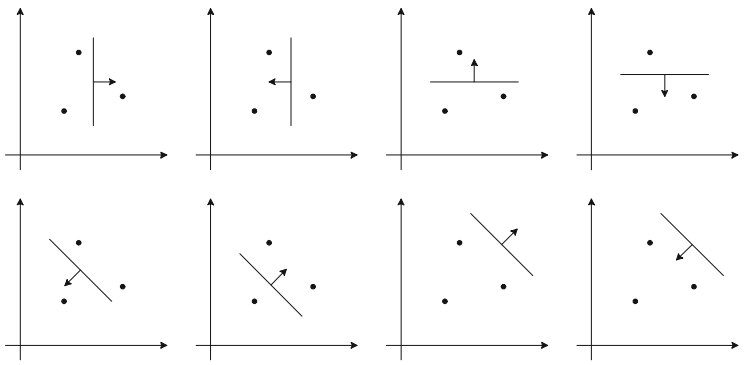

Fig. 20.11 Eight possible ways of shattering 3 points on the plane with a linear indicator function

The SVM in the Linearly Separable Case

First we will describe the SVM in the linearly separable case. The family $\mathcal{F}$ of classification functions in the data space is given by:

$$\mathcal{F} = \{x^\top w + b, w \in \mathbb{R}^p, b \in \mathbb{R}\} \tag{20.19}$$

In order to determine the support vectors we choose $f \in \mathcal{F}$ (or equivalently (w, b)) such that the so-called margin—the corridor between the separating hyperplanes—is maximal. This situation is illustrated in Fig. 20.12. The margin is equal to $d_- + d_+$. The classification function is a hyperplane plus the margin zone, where, in the separable case, no observations can lie. It separates the points from both classes with the highest "safest" distance (margin) between them. It can be shown that margin maximisation corresponds to the reduction of complexity as given by the VC-dimension of the SVM classifier. Apparently, the separating hyperplane is defined only by the *support vectors* that hold the hyperplanes parallel to the separating one. In Fig. 20.12 there are three support vectors that are marked with bold style: two crosses and one circle. We come now to the description of the SVM selection.

Let $x^\top w + b = 0$ be a separating hyperplane. Then d_+ (d_-) will be the shortest distance to the closest objects from the classes $+1$ (-1). Since the separation can be done without errors, all observations $i = 1, 2, \ldots, n$ must satisfy:

$$\begin{aligned} x_i^\top w + b &\geq +1 \quad \text{for} \quad y_i = +1 \\ x_i^\top w + b &\leq -1 \quad \text{for} \quad y_i = -1 \end{aligned}$$

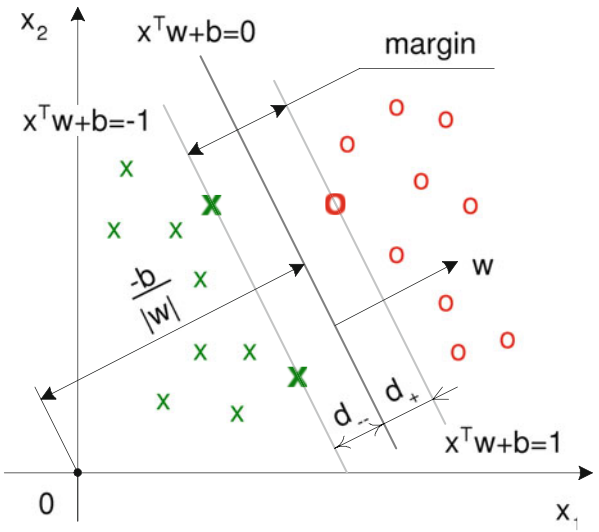

Fig. 20.12 The separating hyperplane $x^\top w + b = 0$ and the margin in the linearly separable case

We can combine both constraints into one:

$$y_i(x_i^\top w + b) - 1 \geq 0 \qquad i = 1, 2, \ldots, n \tag{20.20}$$

The *canonical hyperplanes* $x_i^\top w + b = \pm 1$ are parallel and the distance between each of them and the separating hyperplane is $d_+ = d_- = 1/\|w\|$. To maximise the margin $d_+ + d_- = 2/\|w\|$ one therefore minimises the Euclidean norm $\|w\|$ or its square $\|w\|^2$.

The Lagrangian for the primal problem that corresponds to margin maximisation subject to constraint (20.20) is:

$$L_P(w, b) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \{y_i(x_i^\top w + b) - 1\} \tag{20.21}$$

The Karush–Kuhn–Tucker (KKT) (Gale et al., 1951) first order optimality conditions are:

$$\begin{aligned}
\frac{\partial L_P}{\partial w} = 0: \quad & w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \\
\frac{\partial L_P}{\partial b} = 0: \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\
& y_i(x_i^\top w + b) - 1 \geq 0, \quad i = 1, \ldots, n \\
& \alpha_i \geq 0 \\
& \alpha_i \{y_i(x_i^\top w + b) - 1\} = 0
\end{aligned}$$

From these first order condition, we can derive $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ and therefore the summands in (20.21) read:

$$\begin{aligned}
\frac{1}{2}\|w\|^2 &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j \\
-\sum_{i=1}^{n} \alpha_i \{y_i(x_i^\top w + b) - 1\} &= -\sum_{i=1}^{n} \alpha_i y_i x_i^\top \sum_{j=1}^{n} \alpha_j y_j x_j + \sum_{i=1}^{n} \alpha_i \\
&= -\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n} \alpha_i
\end{aligned}$$

Substituting this into (20.21) we obtain the Lagrangian for the dual problem:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j . \tag{20.22}$$

The primal and dual problems are:

$$\min_{w,b} L_P(w, b)$$

$$\max_{\alpha} L_D(\alpha) \quad \text{s.t.} \quad \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

Since the optimisation problem is convex the dual and primal formulations give the same solution.

Those points i for which the equation $y_i(x_i^\top w + b) = 1$ holds are called support vectors. After "training the SVM" i.e. solving the dual problem above and deriving Lagrange multipliers (they are equal to 0 for non-support vectors) one can classify a company. One uses the classification rule:

$$g(x) = \text{sign}\left(x^\top w + b\right), \tag{20.23}$$

where $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ and $b = \frac{1}{2}(x_{+1} + x_{-1})\, w.\ x_{+1}$ and x_{-1} are two support vectors belonging to different classes for which $y(x^\top w + b) = 1$. The value of the classification function (the score of a company) can be computed as

$$f(x) = x^\top w + b. \tag{20.24}$$

Each score $f(x)$ uniquely corresponds to a default probability (PD). The higher $f(x)$ the higher the PD.

SVMs in the Linearly Non-separable Case

In the linearly non-separable case the situation is like in Fig. 20.13. The slack variables ξ_i represent the violation from strict separation. In this case the following inequalities can be induced from Fig. 20.13:

$$x_i^\top w + b \geq 1 - \xi_i \text{ for } y_i = 1,$$
$$x_i^\top w + b \leq -1 + \xi_i \text{ for } y_i = -1,$$
$$\xi_i \geq 0.$$

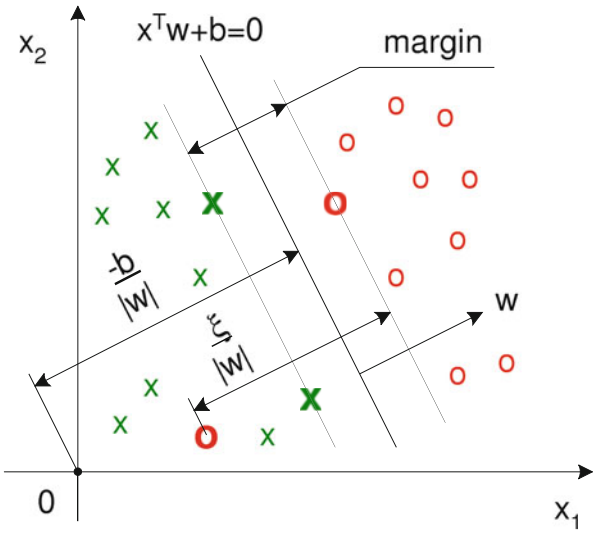

Fig. 20.13 The separating hyperplane $x^\top w + b = 0$ and the margin in the linearly non-separable case

They can be combined into two constraints:

$$y_i (x_i^\top w + b) \geq 1 - \xi_i \tag{20.25}$$

$$\xi_i \geq 0. \tag{20.26}$$

SVM classification again maximises the margin given a family of classification functions $\mathcal{F}$.

The penalty for misclassification, the classification error $\xi_i \geq 0$, is related to the distance from a misclassified point x_i to the canonical hyperplane bounding its class. If $\xi_i > 0$, an error in separating the two sets occurs. The objective function corresponding to penalised margin maximisation is then formulated as:

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i, \tag{20.27}$$

where the parameter C characterises the weight given to the classification errors. The minimisation of the objective function with constraint (20.25) and (20.26) provides the highest possible margin in the case when classification errors are inevitable due to the linearity of the separating hyperplane. Under such a formulation the problem is convex.

The Lagrange function for the primal problem is:

$$L_P (w, b, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \{y_i \left(x_i^\top w + b\right) - 1 + \xi_i\} - \sum_{i=1}^{n} \mu_i \xi_i, \tag{20.28}$$

where $\alpha_i \geq 0$ and $\mu_i \geq 0$ are Lagrange multipliers. The primal problem is formulated as:

$$\min_{w,b,\xi} L_P\,(w, b, \xi)\,.$$

The first order conditions in this case are:

$$\begin{aligned}
\frac{\partial L_P}{\partial w} = 0: &\qquad w - \sum_{i=1}^{n} \alpha_i\, y_i\, x_i = 0 \\
\frac{\partial L_P}{\partial b} = 0: &\qquad \sum_{i=1}^{n} \alpha_i\, y_i = 0 \\
\frac{\partial L_P}{\partial \xi_i} = 0: &\qquad C - \alpha_i - \mu_i = 0
\end{aligned}$$

With the conditions for the Lagrange multipliers:

$$\begin{aligned}
&\alpha_i \geq 0 \\
&\mu_i \geq 0 \\
&\alpha_i \{y_i (x_i^\top w + b) - 1 + \xi_i\} = 0 \\
&\mu_i \xi_i = 0
\end{aligned}$$

Note that $\sum_{i=1}^{n} \alpha_i\, y_i\, b = 0$ therefore similar to the linear separable case the primal problem translates into:

$$\begin{aligned}
L_D\,(\alpha) &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j\, y_i\, y_j\, x_i^\top x_j - \sum_{i=1}^{n} \alpha_i\, y_i\, x_i^\top \sum_{j=1}^{n} \alpha_j\, y_j\, x_j \\
&\quad + C \sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \xi_i - \sum_{i=1}^{n} \mu_i \xi_i \\
&= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j\, y_i\, y_j\, x_i^\top x_j + \sum_{i=1}^{n} \xi_i\,(C - \alpha_i - \mu_i)
\end{aligned}$$

Since the last term is 0 we derive the dual problem as:

$$L_D\,(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j\, y_i\, y_j\, x_i^\top x_j\,, \tag{20.29}$$

and the dual problem is posed as:

$$\max_{\alpha} L_D(\alpha),$$

subject to:

$$0 \leq \alpha_i \leq C,$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

Non-linear Classification

The SVMs can also be generalised to the non-linear case. In order to obtain non-linear classifiers as in Fig. 20.14 one maps the data with a non-linear structure via a function $\Psi : \mathbb{R}^p \mapsto \mathbb{H}$ into a very large dimensional space $\mathbb{H}$ where the classification rule is (almost) linear. Note that all the training vectors x_i appear in L_D (20.29) only as scalar products of the form $x_i^\top x_j$. In the non-linear SVM situations this transforms to $\psi(x_i)^\top \psi(x_j)$.

The so-called *kernel trick* is to compute this scalar product via a kernel function. These kernel functions are actually related to those we presented in Sect. 1.3. If a kernel function K exists such that $K(x_i, x_j) = \Psi(x_i)^\top \Psi(x_j)$, then it can be used without knowing the transformation Ψ explicitly. A necessary and sufficient condition for a symmetric function $K(x_i, x_j)$ to be a kernel is given by Mercer's theorem (Mercer, 1909). It requires positive definiteness, i.e. for

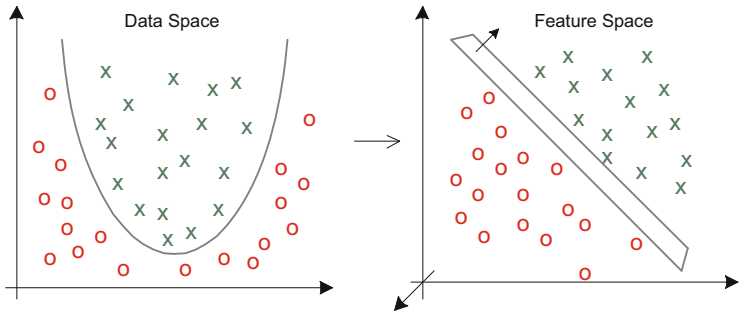

Fig. 20.14 Mapping into a three-dimensional feature space from a two-dimensional data space $\mathbb{R}^2 \mapsto \mathbb{R}^3$. The transformation $\Psi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^\top$ corresponds to the kernel function $K(x_i, x_j) = (x_i^\top x_j)^2$

any data set $x_1, \ldots, x_n$ and any real numbers $\lambda_1, \ldots, \lambda_n$ the function K must satisfy

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j K(x_i, x_j) \geq 0. \tag{20.30}$$

Some examples of kernel functions are:

- $K(x_i, x_j) = e^{-\|x_i - x_j\|/2\sigma^2}$—the isotropic Gaussian kernel with constant σ
- $K(x_i, x_j) = e^{-(x_i - x_j)^\top r^{-2} \Sigma^{-1} (x_i - x_j)/2}$—the stationary Gaussian kernel with an anisotropic radial basis with constant r and variance–covariance matrix Σ from training set
- $K(x_i, x_j) = (x_i^\top x_j + 1)^p$—the polynomial kernel of degree p
- $K(x_i, x_j) = \tanh(k x_i^\top x_j - \delta)$—the hyperbolic tangent kernel with constant k and δ.

SVMs for Simulated Data

The basic parameters of SVMs are on the scaling r of the anisotropic radial basis functions (in the stationary Gaussian kernel) and the capacity C. The parameter r controls the local resolution of the SVM in the sense that smaller r create smaller curvature of the margin. The capacity C controls the amount of slack to allow for unclassified observations. A large C would create a very rough and curved margin where C close to zero makes the margin more smooth.

One of the guinea pig tests for a classification algorithm is the data described as "orange peel", i.e. when two groups of observations have similar means, their variance, however, being different. The classification results in this case are presented in Fig. 20.15. An SVM with a radial basis kernel is highly suitable for such a kind of data.

Another popular non-linear test is the classification of "spiral data". We generated two spirals with the distance between them equal 1.0 that span over 3π radian. The SVM was chosen with $r = 0.1$ and $C = 10/n$. The SVM was able to separate the classes without an error if noise with parameters $\varepsilon_i \sim N(0, 0.1^2 \mathcal{I})$ was injected into the pure spiral data (Fig. 20.16). Obviously, both the "orange peel" and the "spiral data" are not linearly separable.

Solution of the SVM Classification Problem

The standard SVM optimisation problem (20.29), which is a quadratic optimisation problem, is usually solved by means of quadratic programming (QP). This

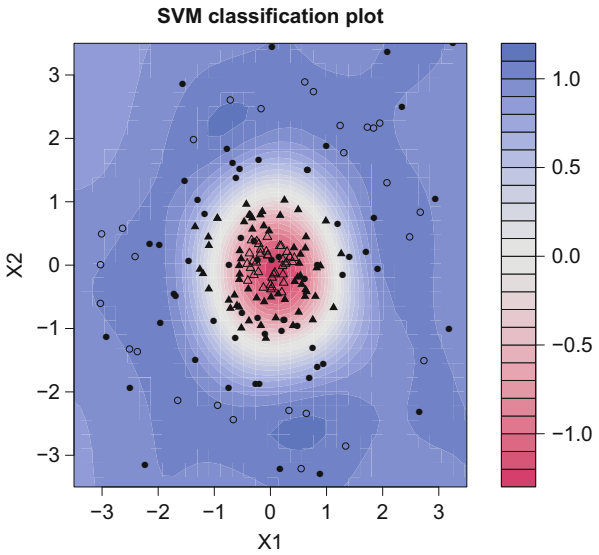

Fig. 20.15 SVM classification results for the "orange peel" data, $n = 200$, $d = 2$, $n_{-1} = n_{+1} = 100$, $x_{+1,i} \sim N((0,0)^\top, 2^2\mathcal{I})$, $x_{-1,i} \sim N((0,0)^\top, 0.5^2\mathcal{I})$ with SVM parameters $r = 0.5$ and $C = 20/200$
`MVAsvmOrangePeel`

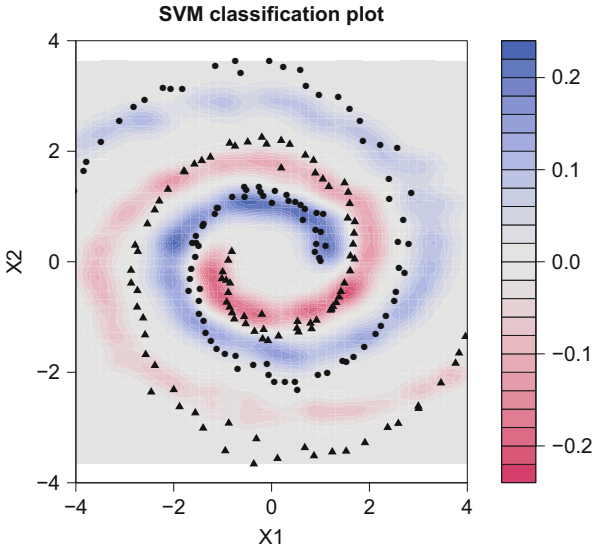

Fig. 20.16 SVM classification results for the noisy spiral data. The spirals spread over 3π radian; the distance between the spirals equals 1.0. $d = 2$, $n_{-1} = n_{+1} = 100$, $n = 200$. The noise was injected with the parameters $\varepsilon_i \sim N(0, 0.1^2\mathcal{I})$. The separation is perfect with SVM parameters $r = 0.1$ and $C = 10/200$
`MVAsvmSpiral`

technique, however, is notorious for (i) its bad scaling properties (the time required to solve the problem is proportional to n^3, where n is the number of observations), (ii) implementation difficulty and (iii) enormous memory requirements. With the QP technique the whole kernel matrix of the size $n \times n$ has to be fit in the memory, which, assuming that each variable takes up 10 bytes of memory, will require $10 \times n \times n$ bytes. This means that 1 million observation (which is not unusual for practical applications such as credit scoring) will require 12,000 TBytes (terabytes)

or 10,000,000 MBytes of operating memory to store. With a typical size of the computer memory of 512 MBytes no more than around 5,000 observations can be processed. Thus, the main emphasis in designing new algorithms was made on using special properties of SVMs to speed up the solution and reduce memory requirements.

Scoring Companies

For our illustration we selected the largest bankrupt companies with the capitalisation of no less than 1 billion USD. The dataset used in this work is from the Credit reform database provided by the Research Data Center (RDC) of the Humboldt Universität zu Berlin. It contains financial information from about 20,000 solvent and 1,000 insolvent German companies. The period spans from 1996 to 2002 and in the case of the insolvent companies the information is gathered 2 years before the insolvency took place. The last annual report of a company before it goes bankrupt receives the indicator $y = 1$ and for the rest (solvent) companies $y = -1$.

We are given 28 variables, i.e. cash, inventories, equity, EBIT, number of employees, and branch code. From the original data, we create common financial indicators which are denoted as $x1, \ldots, x25$. These ratios can be grouped into four categories such as profitability, leverage, liquidity, and activity.

Obviously, data for the year of 1996 are missing and we will exclude them for further calculations. In order to reduce the effect of the outliers on the results, all observations that exceeded the upper limit of IQ (Inter-quartile range) or the lower limit of IQ were replaced with these values. To demonstrate how performance changes, we will use the Accounts Payable (AP) turnover (named $X24$) and ratio of Operating Income (OI) and Total Asset (TA) (named $X3$). We choose randomnly 50 solvent and 50 insolvent companies. The statistical description of financial ratios is summarized in Table 20.5.

Keep in mind that different kernels will influence performance. We will use one of the most common ones, the isotropic Gaussian kernel. Triangles and circles in Fig. 20.17 represent successful and failing companies from the training set, respectively. The coloured background corresponds to different score values f. The more blue the area, the higher the score and the greater the probability of default. Most successful companies lying in the red area have positive profitability and a reasonable activity.

Figure 20.17 presents the classification results for an SVM using isotropic Gaussian kernel with $\sigma = 100$ and the fixed capacity $C = 1$. With given priors, the SVM has trouble classifying between solvent and insolvent company. The radial base σ, which determines the minimum radius of a group, is too large. Notice that SVM do a poor job of distinguishing between groups even though most observations are used as support vector.

The applied SVMs differed in two aspects: (i) their capacity that is controlled by the coefficient C in (20.28) and (ii) the complexity of classifier functions controlled in our case by the isotropic radial basis in the Gaussian kernel. In

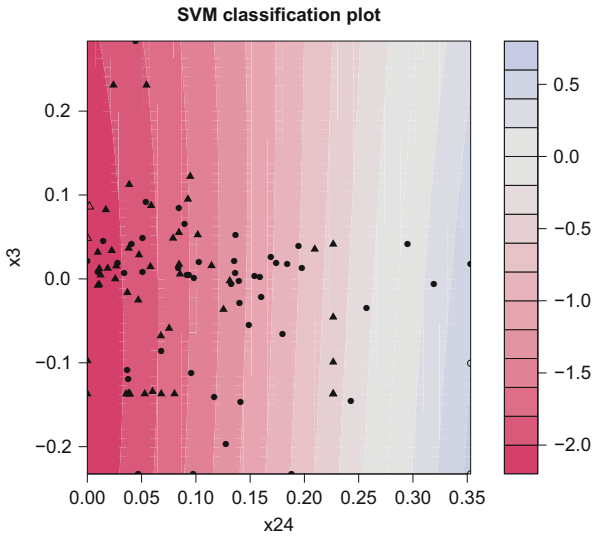

Fig. 20.17 Ratings of companies in two dimensions. Low complexity of classifier functions with $\sigma = 100$ and $C = 1$. Percentage of misclassification is 0.43 `MVAsvmSig100C1`

Table 20.5 Descriptive statistics for financial ratios

Ratio	$q_{0.05}$	Med.	$q_{0.95}$	IQR
OI/TA	−0.22	0.00	0.10	0.06
AP/sales	0.03	0.14	0.36	0.10

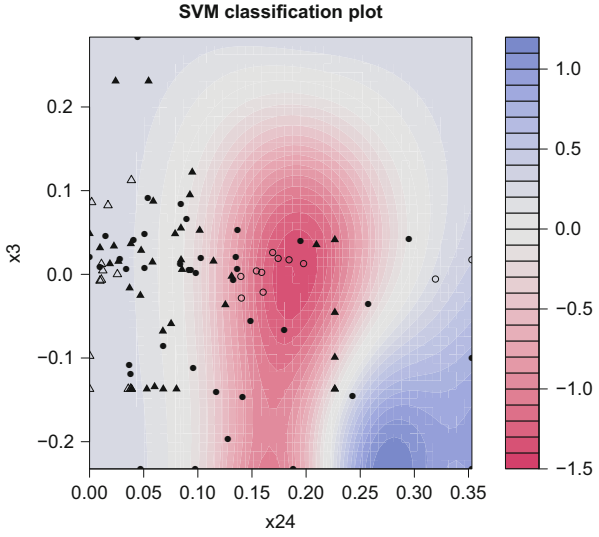

Fig. 20.18 Ratings of companies in two dimensions. The case of an average complexity of classifier functions with $\sigma = 2$ and capacity is fixed at $C = 1$. Percentage of misclassification is reduced to 0.27 `MVAsvmSig2C1`

Fig. 20.18 the value σ is reduced to 2 while C remains the same. SVM start recognising the difference between solvent and insolvent companies resulting in sharper cluster. Figure 20.19 demonstrate the effect of the changing capacity to the classification result. The optimisation of SVM parameters (C and σ) can be done

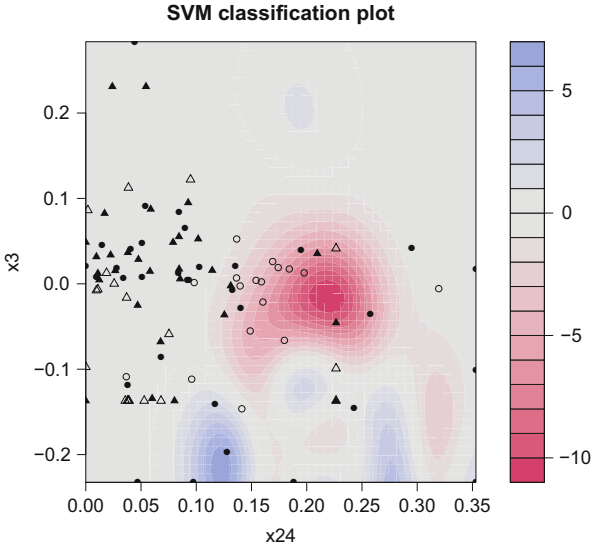

Fig. 20.19 Ratings of companies in two dimensions. High capacity ($C = 200$) with radial basis is fixed at $\sigma = 0.5$. Percentage of misclassification is 0.10 `MVAsvmSig05C200`

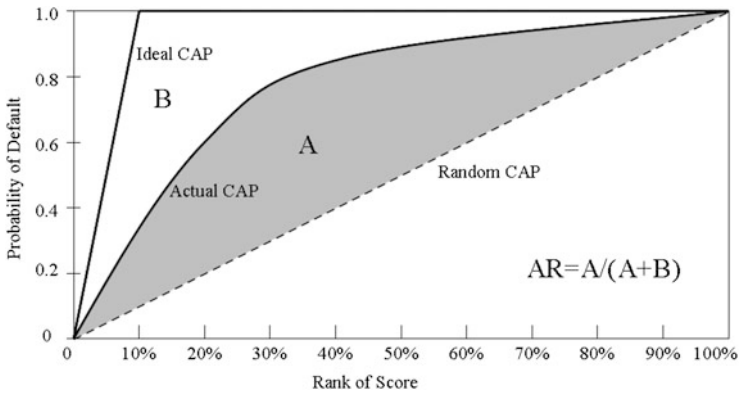

Fig. 20.20 Cumulative accuracy profile (CAP) curve

by using grid search method or an other advance algorithm the so-called Genetic Algorithm.

Figure 20.20 shows a Cumulative Accuracy Profile (CAP) curve which is particularly useful in that it simultaneously measures Type I and Type II errors. In statistical terms, the CAP curve represents the cumulative probability of default events for different percentiles of the risk score scale. Now, we introduce Accuracy Ratio (AR) derived from CAP curve for measuring and comparing the performance of credit risk model. Therefore, AR is defined as the ratio of the area between a model CAP curve and the random curve to the area between the perfect CAP curve

and the random CAP curve (see Fig. 20.20). Perfect classification is attained if the value of AR is equal to one.

Summary

- ↪ SVM classification is done by mapping the data into feature space and finding a separating hyperplane there
- ↪ The support vectors are determined via a quadratic optimisation problem
- ↪ SVM produces highly non-linear classification boundaries

20.5 Classification and Regression Trees

Classification and Regression Trees (CART) is a method of data analysis developed by a group of American statisticians (Breiman et al. , 1984). The aim of CART is to classify observations into a subset of *known classes* or to predict levels of regression functions. CART is a non-parametric tool which is designed to represent decision rules in a form of the so-called *binary trees*. Binary trees split a learning sample parallel to the coordinate axis and represent the resulting data clusters hierarchically starting from a *root node* for the whole learning sample itself and ending with relatively homogenous buckets of observations.

Regression trees are constructed in a similar way but the final buckets do not represent classes but rather approximations to an unknown regression functions at a particular point of the independent variable. In this sense regression trees are estimates via a non-parametric regression model. Here we provide an outlook of how decision trees are created, what challenges arise during practical applications and, of course, a number of examples will illustrate the power of CART.

How Does CART Work?

Consider the example of how high risk patients (those who will not survive at least 30 days after a heart attack is admitted) were identified at San Diego Medical Center, University of California on the basis of initial 24-h data. A classification rule using at most three decisions (questions) is presented in Fig. 20.21. Left branches of the tree represent cases of positive answers, right branches—negative ones so that e.g. if minimum systolic blood pressure over the last 24 h is less or equal 91, then the

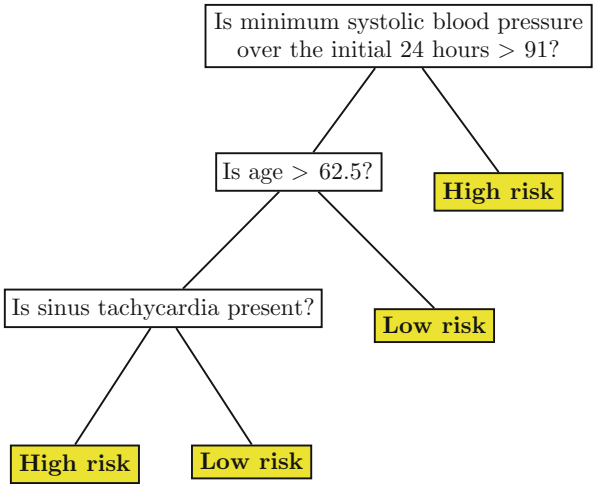

Fig. 20.21 Decision tree for low/high patients

patient belongs to *the high risk* group. In this example the dependant variable is binary: low risk (0) and high risk (1).

A different situation occurs when we are interested in the expected *amount* of days the patient will be able to survive. The decision tree will probably change and the *terminal nodes* will now indicate a mean expected number of days the patient will survive. This situation describes a regression tree rather than a classification tree.

In a more formal setup let Y be a dependent variable—binary or continuous and $X \in \mathbb{R}^d$. We are interested in approximating

$$f(x) = \mathsf{E}(Y|X = x)$$

For the definition of conditional expectations we refer to Sect. 4.2. CART estimates this function f by a step function that is constructed via splits along the coordinate axis. An illustration is given in Fig. 20.22. The regression function $f(x)$ is approximated by the values of the step function. The splits along the coordinate axes are to be determined from the data.

The following simple one-dimensional example shows that the choice of splits points involves some decisions. Suppose that $f(x) = \boldsymbol{I}(x \in [0,1]) + 2\boldsymbol{I}(x \in [1,2])$ is a simple step function with a step at $x = 1$. Assume now that one observes $Y_i = f(x_i) + \varepsilon_i, X_i \sim U[0,2], \varepsilon_i \sim N(0,1)$. By going through the X data points as possible split points one sees that in the neighbourhood of $x = 1$ one has two possibilities: one simply takes the X_i left to 1 or the observation right to 1. In order to make such splits unique one averages these neighbouring points.

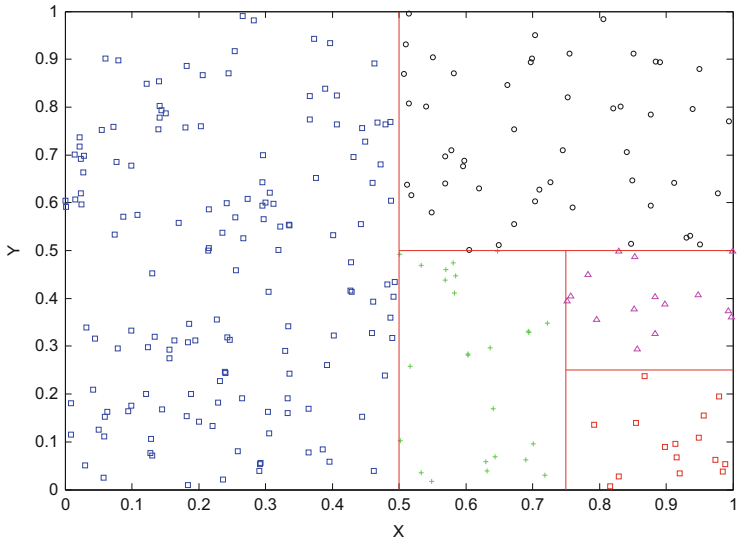

Fig. 20.22 CART orthogonal splitting example where each *colour* corresponds to one cluster

Impurity Measures

A more formal framework on how to split and where to split needs to be developed. Suppose there are n observations in the learning sample and n_j is the overall number of observations belonging to class j, $j = 1, \ldots, J$. The *class probabilities* are:

$$\pi(j) = \frac{n_j}{n}, j = 1, \ldots, J \tag{20.31}$$

$\pi(j)$ is the proportion of observations belonging to a particular class. Let $n(t)$ be the number of observations at node t and $n_j(t)$—the number of observations belonging to the j-th class at t. The frequency of the event that an observation of the j-th class falls into node t is:

$$p(j,t) = \pi(j)\frac{n_j(t)}{n_j} \tag{20.32}$$

The proportion of observations at t are $p(t) = \sum_{j=1}^{J} p(j,t)$ the *conditional probability* of an observation to belong to class j given that it is at node t is:

$$p(j\,|\,t) = \frac{p(j,t)}{p(t)} = \frac{n_j(t)}{n(t)} \tag{20.33}$$

Define now a degree of class homogeneity in a given node. This characteristic—an *impurity measure* $i(t)$—will represent a class homogeneity indicator for a given tree node and hence will help to find optimal splits. Define an *impurity function* $\iota(t)$ which is determined on $(p_1, \ldots, p_J) \in [0,1]^J$ with $\sum_{j=1}^{J} p_j = 1$ so that:

1. ι has a unique maximum at point $\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right)$;
2. ι has a unique minimum at points $(1,0,0,\ldots,0)$, $(0,1,0,\ldots,0)$, …, $(0,0,0,\ldots,1)$;
3. ι is a symmetric function of $p_1, \ldots, p_J$

Each function satisfying these conditions is called an impurity function. Given ι, define the *impurity measure* $i(t)$ for a node t as:

$$i(t) = \iota\{p(1|t), p(2|t), \ldots, p(J|t)\} \tag{20.34}$$

Denote an arbitrary data split by s, then for a given node t which we will call a *parent node* two *child nodes* described in Fig. 20.23 arise: t_L and t_R representing observations meeting and not meeting the split criterion s. A fraction p_L of data from t falls to the left child node and $p_R = 1 - p_L$ is the share of data in t_R.

A *quality measure* of how well split s works is:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R) \tag{20.35}$$

The higher the value of $\Delta i(s,t)$ the better split we have since data impurity is reduced. In order to find an optimal split s it is natural to maximise $\Delta i(s,t)$. Note that in (20.35) for different splits s, the value of $i(t)$ remains constant, hence it is equivalent to find

$$\begin{aligned} s^* &= \underset{s}{\operatorname{argmax}}\ \Delta i\,(s,t) \\ &= \underset{s}{\operatorname{argmax}}\ \{-p_L i\,(t_L) - p_R i\,(t_R)\} \\ &= \underset{s}{\operatorname{argmax}}\ \{p_L i\,(t_L) + p_R i\,(t_R)\} \end{aligned}$$

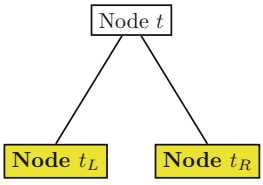

Fig. 20.23 Parent and child nodes hierarchy

where t_L and t_R are implicit functions of s. This splitting procedure is repeated until one arrives at a minimal bucket size. Classes are then assigned to terminal nodes using the following rule:

$$\text{If } p(j\,|\,t) = \max_i p(i\,|\,t), \text{ then } j^*(t) = j \tag{20.36}$$

If the maximum is not unique, then $j^*(t)$ is assigned randomly to those classes for which $p(i\,|\,t)$ takes its maximum value. The crucial question is of course to define an impurity function $i\,(t)$. A natural definition of impurity is via a *variance* measure: Assign 1 to all observations at node t belonging to class j and 0 to others. A sample variance estimate for node t observations is $p(j\,|\,t)\,\{1 - p(j\,|\,t)\}$.

Summing over all J classes we obtain the *Gini index*:

$$i\,(t) = \sum_{j=1}^{J} p(j\,|\,t)\,\{1 - p(j\,|\,t)\} = 1 - \sum_{j=1}^{J} p^2(j\,|\,t) \tag{20.37}$$

The Gini index is an impurity function $\iota(p_1, \ldots, p_J)$, $p_j = p(j\,|\,t)$. It is not hard see that the Gini index is a convex function. Since $p_L + p_R = 1$, we get:

$$\begin{aligned} i(t_L)p_L + i(t_R)p_R &= \iota\,\{p(1|\,t_L), \ldots, p(J\,|\,t_L)\}\,p_L + \iota\,\{p(1|\,t_R), \ldots, p(J\,|\,t_R)\}\,p_R \\ &\le \iota\,\{p_L p(1|\,t_L) + p_R p(1|\,t_R), \ldots, p_L p(J\,|\,t_L) + p_R p(J\,|\,t_R)\} \end{aligned}$$

where inequality becomes an equality in case $p(j\,|\,t_L) = p(j\,|\,t_R)$, $j = 1, \ldots, J$.

Recall that

$$\frac{p(j, t_L)}{p(t)} = \frac{p(t_L)}{p(t)} \cdot \frac{p(j, t_L)}{p(t_L)} = p_L p(j\,|\,t_L)$$

and since

$$p(j\,|\,t) = \frac{p(j, t_L) + p(j, t_R)}{p(t)} = p_L p(j\,|\,t_L) + p_R p(j\,|\,t_R)$$

we can conclude that

$$i(t_L)p_L + i(t_R)p_R \le i(t) \tag{20.38}$$

Hence each variant of data split leads to $\Delta i(s, t) > 0$ unless $p(j\,|\,t_R) = p(j\,|\,t_L) = p(j\,|\,t)$, i.e. when no split decreases class heterogeneity.

Impurity measures can be defined in a number of different ways, for practical applications the so-called *twoing rule* can be considered. Instead of maximising impurity change at a particular node, the twoing rule tries to balance as if the learning sample had only two classes. The reason for such an algorithm is that such

a decision rule is able to distinguish observations between general factors on top levels of the tree and take into account specific data characteristics at lower levels.

If $S = \{1, \ldots, J\}$ is the set of learning sample classes, divide it into two subsets

$$S_1 = \{j_1, \ldots, j_n\}, \text{ and } S_2 = S \backslash S_1$$

All observations belonging to S_1 get dummy class 1, and the rest dummy class 2. The next step is to calculate $\Delta i(s,t)$ for different *s as if there were only two (dummy) classes*. Since actually $\Delta i(s,t)$ depends on S_1, the value $\Delta i(s,t,S_1)$ is maximised. Now apply a *two-step procedure*: first, find $s^*(S_1)$ maximising $\Delta i(s,t,S_1)$ and second, find a *superclass* S_1^* maximising $\Delta i\ \{s^*(S_1), t, S_1\}$. In other words the idea of twoing is to find a combination of superclasses at each node that maximises the impurity increment for two classes.

This method provides one big advantage: it finds the so-called *strategic nodes*, i.e. nodes filtering observations in the way that they are different to the maximum feasible extent. Although applying the twoing rule may seem to be desirable especially for data with a big number of classes, another challenge arises: computational speed. Let's assume that the learning sample has J classes, then a set S can be split into S_1 and S_2 by 2^{J-1} ways. For 11 classes data this will create more than 1,000 combinations. Fortunately the following result helps to reduce drastically the amount of computations.

It can be proven (Breiman et al. , 1984) that in a classification task with two classes and impurity measure $p(1|t)p(2|t)$ for an arbitrary split s a superclass $S_1(s)$ is determined by:

$$S_1(s) = \{j : p(j|t_L) \geq p(j|t_R)\},$$

$$\max_{S_1} \Delta i(s,t,S_1) = \frac{p_L p_R}{4} \left\{ \sum_{j=1}^{J} |p(j|t_L) - p(j|t_R)| \right\}^2 \tag{20.39}$$

Hence the twoing rule can be applied in practice as well as Gini index, although the first criterion works a bit slower.

Gini Index and Twoing Rule in Practice

In this section we look at practical issues of using these two rules. Consider a learning dataset from Salford Systems with 400 observations characterising automobiles: their make, type, colour, technical parameters, age etc. The aim is to build a decision tree splitting different cars by their characteristics based on feasible relevant parameters. The classification tree constructed using the Gini index is given in Fig. 20.24.

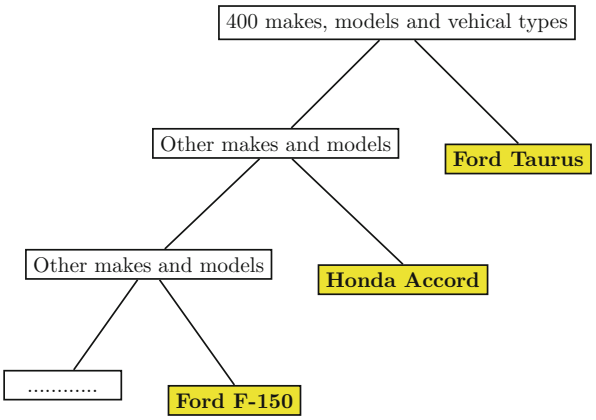

Fig. 20.24 Classification tree constructed by Gini index

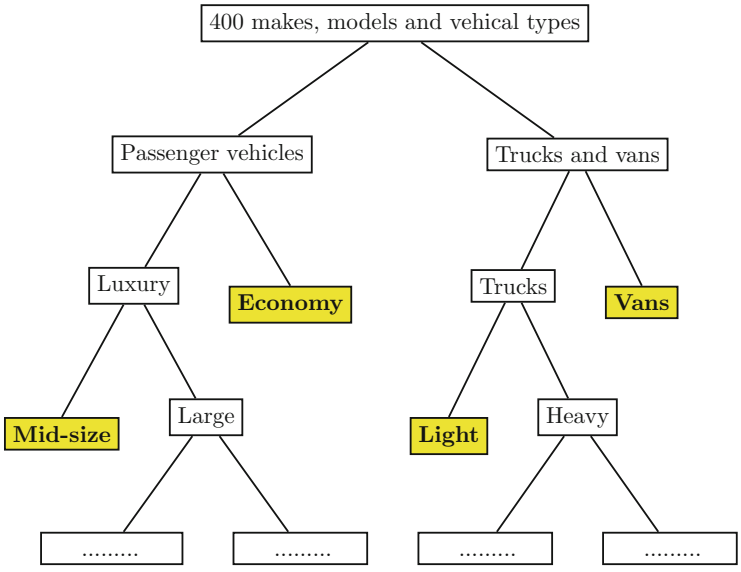

Fig. 20.25 Classification tree constructed by twoing index

A particular feature here is that at each node observations belonging to one make are filtered out, i.e. observations with most striking characteristics are separated. As a result a decision tree is able to pick out automobile makes quite easily.

The twoing rule based tree Fig. 20.25 for the same data is different. Instead of specifying particular car makes at each node, application of the twoing rule results in strategic nodes, i.e. questions which distinguish between different car classes to the maximum extent. This feature can be vital when high-dimensional datasets with a big number of classes are processed.

Optimal Size of a Decision Tree

Up to now we were interested in determining the best split s^* at a particular node. The next and perhaps more important question is how to determine the optimal tree size, i.e. when to *stop splitting*. If each terminal node has only class homogenous dataset, then every point of the learning sample can be flawlessly classified using this *maximum tree*. But can be such an approach fruitful?

The maximum tree is a case of overspecification. Some criterion is required to stop data splitting. Since tree building is dependent on $\Delta i(s,t)$, a criterion is to stop data splitting if

$$\Delta i(s,t) < \bar{\beta} \tag{20.40}$$

where $\bar{\beta}$ is some threshold value.

The value of $\bar{\beta}$ is to be chosen in a subjective way and this is unfortunately a drawback. Empirical simulations show that the impurity increment is frequently non-monotone, that is why even for small $\bar{\beta}$ the tree may be underparametrised. Setting even smaller values for $\bar{\beta}$ will probably remedy the situation but at the cost of tree overparametrisation.

Another way to determine the adequate shape of a decision tree is to demand a minimum number of observations $\overline{N}$ (bucked size) at each terminal node. A disadvantage is that if at terminal node t the number of observations is higher

$$N(t) > \overline{N} \tag{20.41}$$

then this node is also being split as data are still not supposed to be clustered well enough.

Cross-Validation for Tree Pruning

Cross-validation is a procedure which uses the bigger data part as a *training set* and the rest as a *test set*. Then the process is looped so that different parts of the data become learning and training set, so that at the end each datapoint was employed both as a member of test and learning sets. The aim of this procedure is to extract maximum information from the learning sample especially in the situations of data scarceness.

The procedure is implemented in the following way. First, the learning sample is *randomly* divided into V parts. Using the training set from the union of $(V-1)$ subsets a decision tree is constructed while the test set is used to verify the tree quality. This procedure is looped over all possible subsets.

Unfortunately for small values of V cross-validation estimates can be *unstable* since each iteration a cluster of data is selected *randomly* and the number of

iterations itself is relatively small, thus the overall estimation result is somewhat random. Nowadays cross-validation with $V = 10$ is an industry standard and for many applications a good balance between computational complexity and statistical precision.

Cost-Complexity Function and Cross-Validation

Another method taken into account is *tree complexity*, i.e. the *number of terminal nodes*. The maximum tree will get a penalty for its big size, on the other hand it will be able to make perfect in-sample predictions. Small trees will, of course, get lower penalty for their size but their prediction abilities are limited. Optimisation procedure based on such a trade-off criterion could determine a good decision tree.

Define *the internal misclassification error* of an arbitrary observation at node t as $e(t) = 1 - \max_j p(j \mid t)$, define also $E(t) = e(t)p(t)$. Then *internal misclassification tree error* is $E(T) = \sum_{t \in \tilde{T}} E(t)$ where $\tilde{T}$ is a set of terminal nodes. The estimates are called *internal* because they are based solely on the learning sample. It may seem that $E(T)$ as a tree quality measure is sufficient but unfortunately it is not so. Consider the case of the maximum tree, here $E(T_{\mathrm{MAX}}) = 0$, i.e. the tree is of best configuration.

For any subtree $T\ (\leq T_{\mathrm{MAX}})$ define the number of terminal nodes $\left|\tilde{T}\right|$ as a measure of its complexity. The following cost-complexity function can be used:

$$E_\alpha(T) = E(T) + \alpha \left|\tilde{T}\right| \tag{20.42}$$

where $\alpha \geq 0$ is a complexity parameter and $\alpha \left|\tilde{T}\right|$ is a cost component. The more complex the tree (high number of terminal nodes) the lower is $E(T)$ but at the same time the higher is the penalty $\alpha \left|\tilde{T}\right|$ and vice versa.

The number of subtrees of T_{MAX} is finite. Hence pruning of T_{MAX} leads to creation of a subtree sequence $T_1, T_2, T_3, \ldots$ with a decreasing number of terminal nodes.

An important question is if a subtree $T \leq T_{\mathrm{MAX}}$ for a given α minimising $E_\alpha(T)$ always exists and whether it is unique?

In Breiman et al. (1984) it is shown that for $\forall \alpha \geq 0$ there exists an optimal tree $T(\alpha)$ in the sense that

1. $E_\alpha \{T(\alpha)\} = \min_{T \leq T_{\mathrm{MAX}}} E_\alpha(T) = \min_{T \leq T_{\mathrm{MAX}}} \left\{E(T) + \alpha \left|\tilde{T}\right|\right\}$
2. if $E_\alpha(T) = E_\alpha \{T(\alpha)\}$ then $T(\alpha) \leq T$.

This result is a proof of existence, but also a proof of uniqueness: consider another subtree T' so that T and T' both minimise E_α and are not nested, then $T(\alpha)$ does not exist in accordance with second condition.

The idea of introducing cost-complexity function at this stage is to check only a subset of different subtrees of T_{MAX}: optimal subtrees for different values of α. The starting point is to define the first optimal subtree in the sequence so that $E(T_1) = E(T_{\text{MAX}})$ and the size of T_1 is minimum among other subtrees with the same cost level. To get T_1 out of T_{MAX} for each terminal node of T_{MAX} it is necessary to verify the condition $E(t) = E(t_L) + E(t_R)$ and if it is fulfilled—node t is pruned. The process is looped until no extra pruning is available—the resulting tree $T(0)$ becomes T_1.

Define a node t as an *ancestor* of t' and t' as *descendant* of t if there is a connected path down the tree leading from t to t'. Consider Fig. 20.26 where nodes $t_4, t_5, t_8, t_9, t_{10}$ and t_{11} are descendants of t_2 while nodes t_6 and t_7 are not descendants of t_2 although they are positioned lower since it is not possible to connect them with a path from t_2 to these nodes without engaging t_1. Nodes t_4, t_2 and t_1 are ancestors of t_9 and t_3 is not ancestor of t_9.

Define the *branch* T_t of the tree T as a subtree based on node t and all its descendants. An example is given in Fig. 20.27. Pruning a branch T_t from a tree T means deleting all descendant nodes of t. Denote the transformed tree as $T - T_t$. Pruning the branch T_{t_2} results in the tree described in Fig. 20.28.

For any branch T_t define the *internal misclassification estimate* as:

$$E(T_t) = \sum_{t' \in \tilde{T}_t} E(t') \tag{20.43}$$

where $\tilde{T}_t$ is the set of terminal nodes of T_t. Hence for an arbitrary node t of T_t:

$$E(t) > E(T_t) \tag{20.44}$$

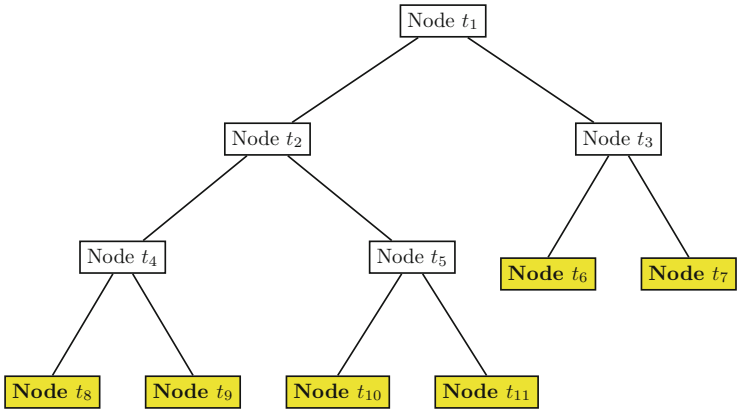

Fig. 20.26 Decision tree hierarchy

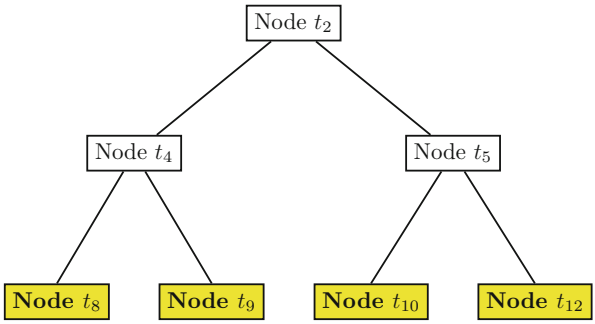

Fig. 20.27 The branch T_{t_2} of the original tree T

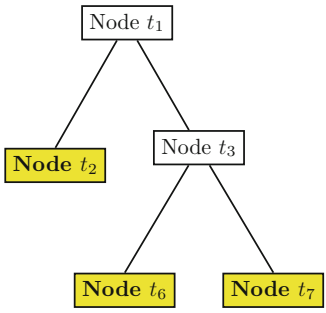

Fig. 20.28 $T - T_{t_2}$ the pruned tree T

Consider now the *cost-complexity misclassification estimate* for branches or single nodes. Define for a single node $\{t\}$:

$$E(\{t\}) = E(t) + \alpha \tag{20.45}$$

and for a branch:

$$E_\alpha(T_t) = E(T_t) + \alpha \left|\tilde{T}_t\right| \tag{20.46}$$

When $E_\alpha(T_t) < E_\alpha(\{t\})$ the branch T_t is preferred to a single node $\{t\}$ according to cost-complexity. For some α both (20.45) and (20.46) will become equal. This critical value of α can be determined from:

$$E_\alpha(T_t) < E_\alpha(\{t\}) \tag{20.47}$$

which is equivalent to

$$\alpha < \frac{E(t) - E(T_t)}{\left|\tilde{T}_t\right| - 1} \tag{20.48}$$

where $\alpha > 0$ since $E(t) > E(T_t)$

To obtain the next member of the subtrees sequence, i.e. T_2 out of T_1 a special node called *weak link* is determined. For this purpose a function $g_1(t)$, $t \in T_1$ is defined as

$$g_1(t) = \begin{cases} \frac{E(t)-E(T_t)}{|\tilde{T}_t|-1}, & t \notin \tilde{T}_1 \\ +\infty, & t \in \tilde{T}_1 \end{cases} \tag{20.49}$$

Node $\bar{t}_1$ is a weak link in T_1 if

$$g_1(\bar{t}_1) = \min_{t \in T_1} g_1(t) \tag{20.50}$$

and a new value for α_2 is defined as

$$\alpha_2 = g_1(\bar{t}_1) \tag{20.51}$$

A new tree $T_2 \prec T_1$ in the sequence is obviously defined by pruning the branch $T_{\bar{t}_1}$, i.e.

$$T_2 = T_1 - T_{\bar{t}_1} \tag{20.52}$$

The process is looped until root node $\{t_0\}$—the final member of sequence—is reached. When there are multiple weak links detected, for instance $g_k(\bar{t}_k) = g_k(\bar{t}'_k)$, then both branches are pruned, i.e. $T_{k+1} = T_k - T_{\bar{t}_k} - T_{\bar{t}'_k}$.

In this way it is possible to get the sequence of optimal subtrees $T_{\text{MAX}} \succ T_1 \succ T_2 \succ T_3 \succ \cdots \succ \{t_0\}$ for which it is possible to prove that the sequence $\{\alpha_k\}$ is increasing, i.e. $\alpha_k < \alpha_{k+1}$, $k \geq 1$ and $\alpha_1 = 0$. For $k \geq 1$: $\alpha_k \leq \alpha < \alpha_{k+1}$ and $T(\alpha) = T(\alpha_k) = T_k$.

Practically this tells us how to implement the search algorithm. First, the maximum tree T_{MAX} is taken, then T_1 is found and a weak link $\bar{t}_1$ is detected and branch $T_{\bar{t}_1}$ is pruned off, α_2 is calculated and the process is continued.

When the algorithm is applied to T_1, the number of pruned nodes is usually quite significant. For instance, consider the following typical empirical evidence (see Table 20.6). When the trees become smaller, the difference in the number of terminal nodes also gets smaller.

Finally, it is worth mentioning that the sequence of optimally pruned subtrees is a subset of trees which might be constructed using direct method of internal misclassification estimator minimisation given a fixed number of terminal nodes.

Table 20.6 Typical pruning speed

Tree	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
$\lvert\tilde{T}_k\rvert$	71	63	58	40	34	19	10	9	7	6	5	2	1

Consider an example of tree $T(\alpha)$ with 7 terminal nodes, then there is no other subtree T with 7 terminal nodes having lower $E(T)$. Otherwise

$$E_\alpha(T) = E(T) + 7\alpha < E_\alpha\{T(\alpha)\} = \min_{T \le T_{\text{MAX}}} E_\alpha(T)$$

which is impossible by definition.

Applying the method of V-fold cross-validation to the sequence $T_{\text{MAX}} \succ T_1 \succ T_2 \succ T_3 \succ \cdots \succ \{t_0\}$, an *optimal tree* is determined. On the other hand it is frequently pointed out that choice of tree with minimum value of $E^{\text{CV}}(T)$ is not always adequate since $E^{\text{CV}}(T)$ is not too robust, i.e. there is a whole range of values $E^{\text{CV}}(T)$ satisfying $E^{\text{CV}}(T) < E^{\text{CV}}_{\text{MIN}}(T) + \varepsilon$ for small $\varepsilon > 0$. Moreover, when $V < N$ a simple change of random generator seed will definitely result in changed values of $\left|\tilde{T}_k\right|$ minimising $\hat{E}(T_K)$. Hence a so-called *one standard error* empirical rule is applied which states that if T_{k_0} is the tree minimising $E^{\text{CV}}(T_{k_0})$ from the sequence $T_{\text{MAX}} \succ T_1 \succ T_2 \succ T_3 \succ \cdots \succ \{t_0\}$, then a value k_1 and a correspondent tree T_{k_1} are selected so that

$$\underset{k_1}{\text{argmax}}\ \hat{E}(T_{k_1}) \le \hat{E}(T_{k_0}) + \sigma\left\{\hat{E}(T_{k_0})\right\} \tag{20.53}$$

where $\sigma(\cdot)$ denotes sample estimate of standard error and $\hat{E}(\cdot)$—the relevant sample estimators.

The dotted line in Fig. 20.29 shows the area where the values of $\hat{E}(T_k)$ only slightly differ from $\min_{\left|\tilde{T}_k\right|} \hat{E}(T_k)$. The left edge which is roughly equivalent to 16 terminal nodes shows the application of one standard error rule. The use of one

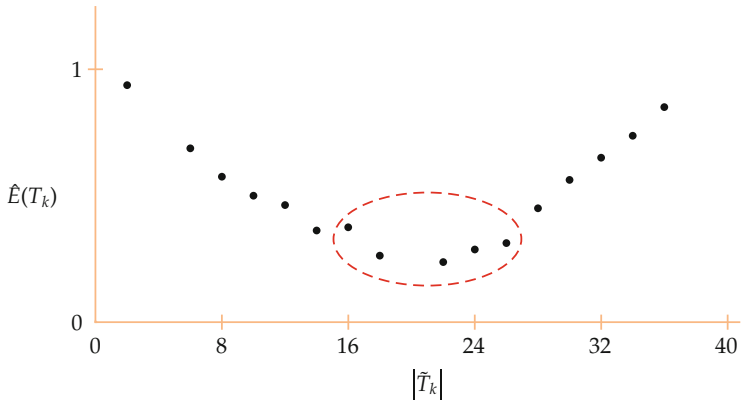

Fig. 20.29 The example of relationship between $\hat{E}(T_k)$ and number of terminal nodes

standard error rule allows not only to achieve more robust results but also to get trees of lower complexity given the error comparable with $\min_{|\tilde{T}_k|} \hat{E}(T_k)$.

Regression Trees

Up to now we concentrate on classification trees. Although *regression trees* share a similar logical framework, there are some differences which need to be addressed. The important difference between classification and regression trees is the type of dependent variable Y. When Y is discrete, a decision tree is called a classification tree, a regression tree is a decision tree with a *continuous* dependent variable.

Gini index and twoing rule discussed in previous sections assume that the number of classes is finite and hence introduce some measures based mainly on $p(j|t)$ for arbitrary class j and node t. But since in case of continuous dependent variable there are no more classes, this approach cannot be used anymore unless groups of continuous values are effectively substituted with artificial classes. Since there are no classes anymore—how can be the maximum regression tree determined? Analogously with discrete case, absolute homogeneity can be then described only after some adequate impurity measure for regression trees is introduced.

Recall the idea of *Gini index*, then it becomes quite natural to use the *variance* as impurity indicator. Since for each node data variance can be easily computed, then splitting criterion for an arbitrary node t can be written as

$$s^* = \underset{s}{\operatorname{argmax}} \; [p_L \operatorname{var}\{t_L(s)\} + p_R \operatorname{var}\{t_R(s)\}] \tag{20.54}$$

where t_L and t_R are emerging child nodes which are, of course, directly dependent on the choice of s^*.

Hence the maximum regression tree can be easily defined as a structure where each node has only the same predicted values. It is important to point out that since continuous data have much higher chances to take different values comparing with discrete ones, the size of maximum regression tree is usually very big.

When the maximum regression tree is properly defined, it is then of no problem to get an optimally size tree. Like with classification trees, maximum regression tree is usually supposed to be upwardly pruned with the help of cost-complexity function and cross-validation. That is why the majority of results presented above is applied to regression trees as well.

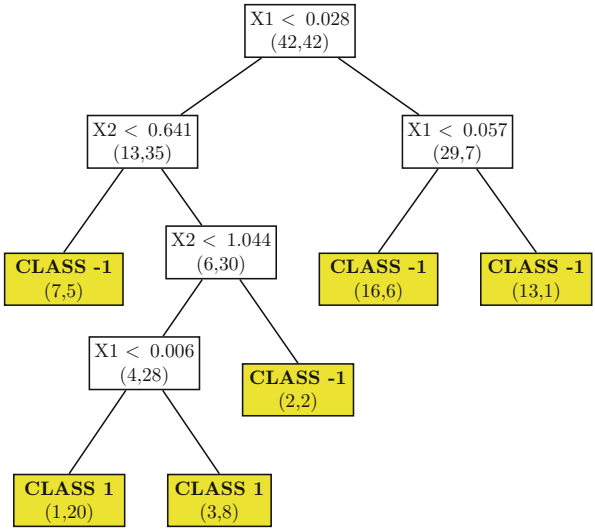

Fig. 20.30 Decision tree for bankruptcy dataset: Gini index, $\bar{N} = 30$
MVACARTBan1

Bankruptcy Analysis

This section provides a practical study on bankruptcy data involving decision trees. A dataset with 84 observations representing different companies is constituted by three variables:

- net income to total assets ratio
- total liabilities to total assets ratio
- company status (-1 if bankrupt and 1 if not)

The data is from SEC (2004).

The goal is to predict and describe the company status given the two primary financial ratios. Since no additional information like the functional form of possible relationship is available, the use of a *classification tree* is an active alternative.

The tree given in Fig. 20.30 was constructed using the Gini index and a $\bar{N} = 30$ constraint, i.e. the number of points in each of the terminal nodes can not be more than 30. Numbers in parentheses displayed on terminal nodes are observation quantities belonging to Class 1 and Class -1.

If we loose the constraint to $\bar{N} = 10$, the decision rule changes, see Fig. 20.31. How exactly did the situation change? Consider the Class 1 terminal nodes of the tree on Fig. 20.30. The first one contains 21 observations and thus was split for $\bar{N} = 10$. When it was split two new nodes of *different classes* emerged and for both of them the impurity measure has decreased.

We may conclude that $\bar{N} \approx 10$ is a good choice and analysing the tree produced we can state that for this particular example the net income to total assets (X_1) ratio appears to be an important class indicator. The successful classification ratio

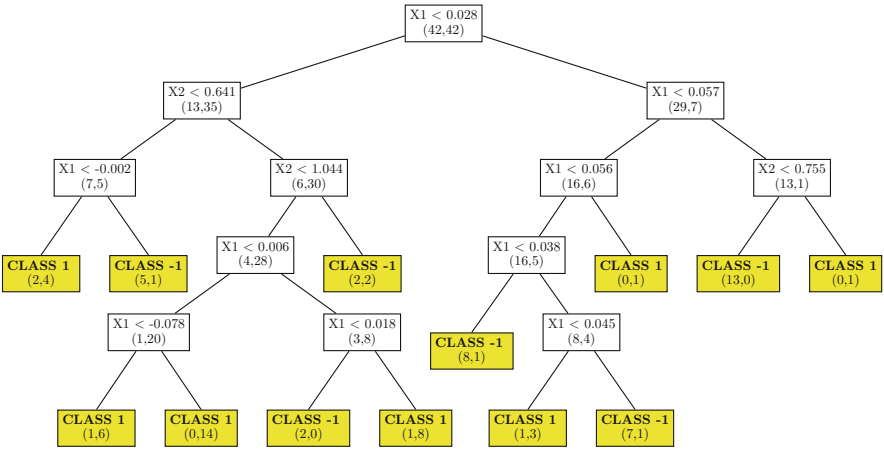

Fig. 20.31 Decision tree for bankruptcy dataset: Gini index, $\bar{N} = 10$ MVACARTBan2

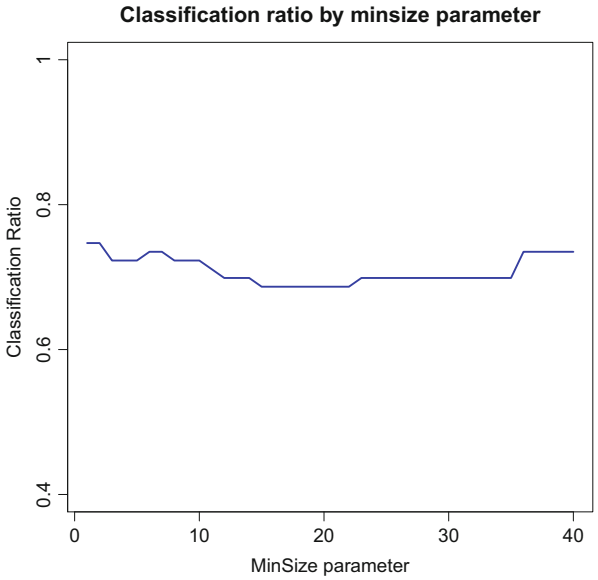

Fig. 20.32 Successful classification ratio dynamic over the number of terminal nodes: cross-validation MVAbancrupcydis

dynamic over the number of terminal nodes is shown in Fig. 20.32. It is chosen by cross-validation method.

For this example with relatively small sample size we construct two maximum trees—using the Gini and twoing rules, see Figs. 20.33 and 20.34. Looking at both decision trees we see that the choice of impurity measure is not so important as the right choice of tree size.

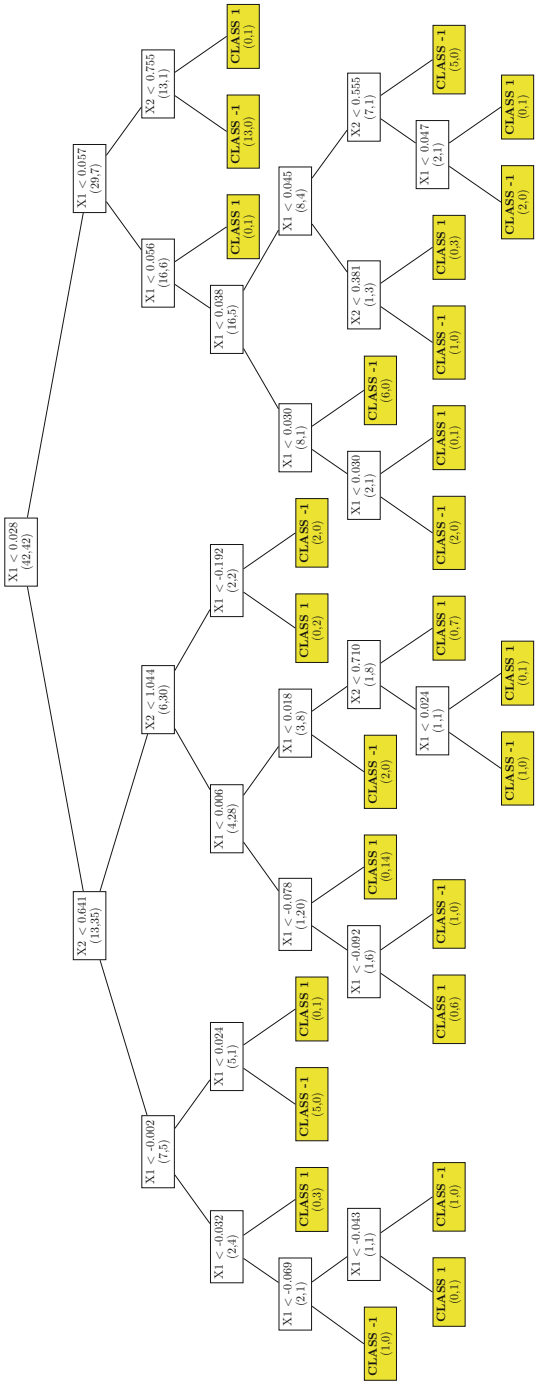

Fig. 20.33 Maximum tree constructed employing Gini index `MVACARTGiniTree1`

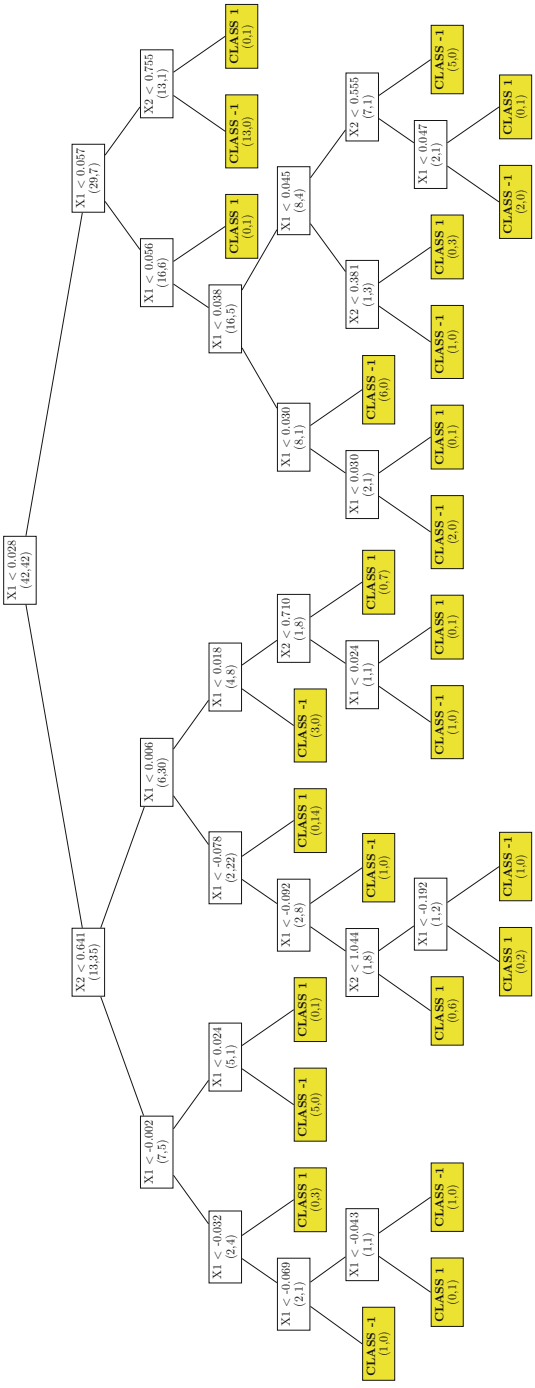

Fig. 20.34 Maximum tree constructed employing twoing rule `MVACARTTwoingTree1`

Summary

- ↪ CART is a tree based method splitting the data sequentially into a binary tree
- ↪ CART determined the nodes by minimising an impurity measure at each mode
- ↪ CART is non-parametric:
  When no data structure hypotheses are available, non-parametric analysis becomes the single effective data mining tool. CART is a flexible nonparametric data mining tool
- ↪ CART does not require variables to be selected in advance:
  From a learning sample CART will automatically select the most significant ones
- ↪ CART is very efficient in computational terms:
  Although all possible data splits are analysed, the CART architecture is flexible enough to do all of them quickly
- ↪ CART is robust to the effect of outliers:
  Due to data-splitting nature of decision rules creation it is possible to distinguish between datasets with different characteristics and hence to neutralise outliers in separate nodes
- ↪ CART can use any combination of continuous and categorical data:
  Researchers are no longer limited to a particular class of data and will be able to capture more real-life examples

20.6 Boston Housing

Coming back to the Boston Housing data set, we compare the results of EPP on the original data $\mathcal{X}$ and the transformed data $\hat{\mathcal{X}}$ motivated in Sect. 1.9. So we exclude X_4 (indicator of Charles River) from the present analysis.

The aim of this analysis is to see from a different angle whether our proposed transformations yield more normal distributions and whether it will yield data with less outliers. Both effects will be visible in our projection pursuit analysis.

We first apply the Jones and Sibson index to the non-transformed data with 50 randomly chosen 13-dimensional directions. Figure 20.35 displays the results in the following form. In the lower part, we see the values of the Jones and Sibson index. It should be constant for 13-dimensional normal data. We observe that this is clearly not the case. In the upper part of Fig. 20.35 we show the standard normal density as a green curve and two densities corresponding to two extreme index values. The red, slim curve corresponds to the maximal value of the index among the 50 projections. The blue curve, which is close to the normal, corresponds to the minimal value of

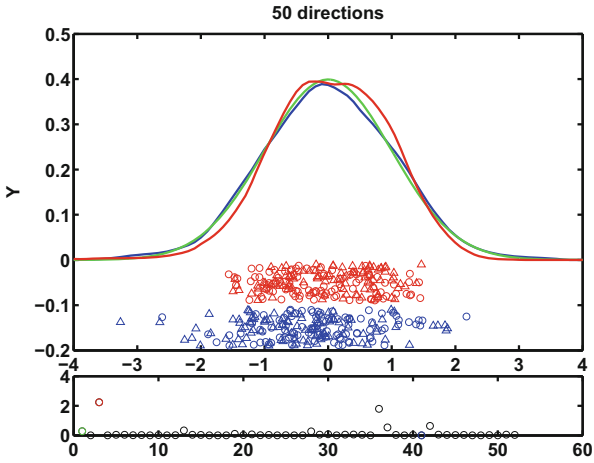

Fig. 20.35 Projection Pursuit with the Sibson–Jones index with 13 original variables
MVAppsib

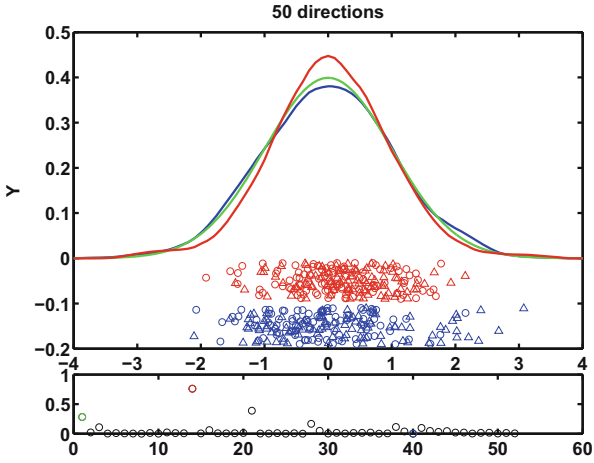

Fig. 20.36 Projection Pursuit with the Sibson–Jones index with 13 transformed variables
MVAppsib

the Jones and Sibson index. The corresponding values of the indices have the same colour in the lower part of Fig. 20.35. Below the densities, a jitter plot shows the distribution of the projected points $\alpha^\top x_i$ $(i = 1, \ldots, 506)$. We conclude from the outlying projection in the red distribution that several points are in conflict with the normality assumption.

Figure 20.36 presents an analysis with the same design for the transformed data. We observe in the lower part of the figure values that are much lower for the Jones and Sibson index (by a factor of 10) with lower variability which suggests that the transformed data is closer to the normal. ("Closeness" is interpreted here in the sense of the Jones and Sibson index.) This is confirmed by looking to the upper part of Fig. 20.36 which has a significantly less outlying structure than in Fig. 20.35.

20.7 Exercises

Exercise 20.1 *Calculate the Simplicial Depth for the Swiss bank notes data set and compare the results to the univariate medians. Calculate the Simplicial Depth again for the genuine and counterfeit bank notes separately.*

Exercise 20.2 *Construct a configuration of points in* $\mathbb{R}^2$ *such that* $x_{\text{med},j}$ *from (20.2) is not in the "centre" of the scatterplot.*

Exercise 20.3 *Apply the SIR technique to the US companies data with* $Y =$ *market value and* $X =$ *all other variables. Which directions do you find?*

Exercise 20.4 *Simulate a data set with* $X \sim N_4(0, I_4)$, $Y = (X_1 + 3X_2)^2 + (X_3 - X_4)^4 + \varepsilon$ *and* $\varepsilon \sim N(0, (0.1)^2)$. *Use SIR and SIR II to find the EDR directions.*

Exercise 20.5 *Apply the Projection Pursuit technique on the Swiss bank notes data set and compare the results to the PC analysis and the Fisher discriminant rule.*

Exercise 20.6 *Apply the SIR and SIR II technique on the car data set in Table 22.3 with* $Y =$ *price.*

Exercise 20.7 *Generate four regions on the two-dimensional unit square by sequentially cutting parallel to the coordinate axes. Generate 100 two-dimensional Uniform random variables and label them according to their presence in the above regions. Apply the CART algorithm to find the regions bound and to classify the observations.*

Exercise 20.8 *Modify Exercise 20.7 by defining the regions as lying above and below the main diagonal of the unit square. Make a CART analysis and comment on the complexity of the tree.*

Exercise 20.9 *Apply the SVM with different radial basis parameter* r *and different capacity parameter* c *in order to separate two circular datasets. This example is often called the Orange Peel exercise and involves two Normal distributions* $N(\mu, \Sigma_i)$, $i = 1, 2$, *with covariance matrices* $\Sigma_1 = 2\mathcal{I}_2$ *and* $\Sigma_2 = 0.5\mathcal{I}_2$.

Exercise 20.10 *The noisy spiral data set consists of two intertwining spirals that need to be separated by a non-linear classification method. Apply the SVM with different radial basis parameter* r *and capacity parameter* c *in order to separate the two spiral datasets.*

Exercise 20.11 *Apply the SVM to separate the bankrupt from the surviving (profitable) companies using the profitability and leverage ratios given in the Bankruptcy data set in Table 22.21.*